Contents

1 Introduction **1**
1.1 Executive Summary 1
1.2 Background 1
1.3 State of The Art 2

2 Technical Documentation **4**
2.1 Application Overview 4
2.1.1 Initial Research Phase 4
2.1.2 Persona Development 7
2.1.3 Design Decision Framework 9
2.1.4 Feature Prioritization 11
2.1.5 Key User Requirements 12
2.1.6 UI Design 12
2.1.7 Literature Survey on AI research 15
2.1.8 Architecture 19
2.1.9 Core System Components 33
2.2 Methods 35
2.2.1 Development Framework 35
2.2.2 Development Stack 36
2.2.3 GUI 37
2.2.4 Data Collection and Processing 39
2.2.5 Model A - CV Analysis System 44
2.2.6 Model B - Job Matching System 48
2.2.7 Model C - Interview Preparation System 52
2.2.8 Model D - Vision Recognition System 55
2.3 Deployment 59
2.3.1 Dockerization of Components 59
2.3.2 LLM Deployment Strategy 59

3 Result and Discussion **64**
3.1 Model A - CV Analysis System 64
3.1.1 Quantitative Performance Metrics 64
3.1.2 Qualitative Analysis 65
3.1.3 Comparison with Baseline Approaches 66
3.1.4 Discussion: Trade Offs and Implications 67
3.1.5 Limitations and Future Work 68
3.2 Model B - Job Matching System 68
3.3 Model C - Interview Preparation System 68
3.4 Model D - Vision Recognition System 68

4 Process Documentation **73**
4.1 Project Management 73
4.2 Timeline and Milestones 75
4.2.1 Project Initiation and Planning (Weeks 1- 3) 75
4.2.2 Research and System Design (Weeks 4 - 6) 75
4.2.3 Prototype Development (Weeks 7-10) 76
4.2.4 Finalization and Documentation (Weeks 11-12) 76

5 Conclusions **77**

A Work Contract **82**

1. Introduction

1.1 Executive Summary

This report presents the design and development of a personalized career coach application leveraging the power of artificial intelligence. The application is designed to help recent graduates in technology to land an IT job by helping them analyze their resume, preparing them for interviews, recommending relevant jobs based on the skills of the individual and has a visual feature as well which helps them identify the companies they are interested in. The core of the application utilizes LLMs (Large Language Models) as its backbone and adheres to the following constraints:

- Operability on small, low power devices (Raspberry Pi farm in this case)
- Multimodal capabilities – one of the features of the application must be able to process audio or visual data and text
- Mixture of Experts – there should be at least 3 different experts who are specialized in different fields
- Fine-tuned LLMs – each expert must be fine-tuned for a designated purpose

The report also explores technical topics such as latest generation of sLLM (Small Large Language Model), training techniques for fine-tuning LLMs, training image models as well as team management topics such as collaboration within teams, defining achievable goals, and project management frameworks. Key challenges include managing a big team working on an abstract idea, moving towards a common goal and technical challenges such as optimizing LLMs to run on distributed low power hardware. Overall, this project focuses on the practical application of state-of-the-art LLMs and fine-tuning techniques to solve a real-world problem along with the challenges faced along the way while creating a product from scratch.

1.2 Background

For newly graduated students entering the job market can be a difficult process. It involves many steps with finding relevant jobs and also find jobs that are suitable based on experience, skill set, and interests. With high competition and an overwhelming amount of job postings it can be hard to find a suitable job.
With a more personalized career coach, the search and application process for a job can become easier and improve the confidence and success rate of the job seeker. It can improve the job seeker's CV, suggest improvements and highlight skills needed for certain positions. Then give the job seeker more tailored recommendations based on the resume.

Furthermore, it can help the user better prepare for interviews with mock interviews and reduce the anxiety a job seeker might experience when attending an interview.
There are some existing solutions like e.g. Finn, Arbeidsplassen and LinkedIn, which help with looking and searching through different job postings. Some of these solutions are good for browsing the current job listings that are posted, and some of them also offer some sort of resume builder. These solutions are limited to the manual process of looking through all the job postings and filtering out unwanted listings and so on. There are also lesser-known services that offer a more fleshed out and personalized AI career coach. These focus mostly on the improvement of users resumes.
With advancements in modern technology, the feasibility of this project is indisputable. The capabilities of current AI models, particularly large language models (LLMs), make it possible to generate highly personalized recommendations for job seekers. By leveraging data scraping techniques to aggregate job listings and employing fine-tuned LLMs to match job descriptions with candidate resumes, the project can deliver precise job-role alignments. Additionally, integrating chatbot technology enables the creation of an interactive interview preparation tool, further enhancing the user's readiness and confidence.
With this in mind, the goal is to create a career coach that can help job seekers identify suitable job opportunities, as well as provide tailored advice for resume optimization. Along with offering interview preparation tips. This would help newly graduated find a suitable job with a more smooth and easy experience.

1.3 State of The Art

Large language models like ChatGPT and Claude have cemented their position in almost every aspect of today's society. LLMs are being utilized in unique and imaginative ways to improve application performance and user experience. Therefore, there has been a sharp rise in "personalized assistant" applications leveraging the power of LLMs including the field of job hunting. Companies such as Teal can build your resume and cover letters with AI, aragon.ai can modify your professional headshots, yoodli can help you prep for interviews and many more [1] [2] [3]. A review of similar applications in the current domain is given below:

Most existing applications that provide similar functionalities do not provide fully personalized or multimodal advice. The systems process includes CV optimization and job searching. The platform "LinkedIn" offers job recommendations based on profile information but no guidance for individual career paths. The "Zety Resume Builder" optimizes resumes by using AI but does not have an option for advanced job matching.

Multimodal AI models like Anthropic's Claude 3.5 Sonnet or OpenAI's GPT-4 version can integrate text and image inputs for holistic analysis. These techniques include cross-modal attention mechanisms and transformers for multimodal to enable the analysis of resumes (PDFs), as well as video interviews and audio inputs.

Platforms like Duolingo use AI to provide personalized language learning paths. Coursera uses AI for personalized and interactive learning for their users.

The strengths of current AI approaches in career guidance are that they can approach their ability to deliver scalable and accessible solutions. The system can analyze the job pro-

files and candidate skills with remarkable speed and significantly reduce the time it takes to match preferences.However, all of the above mentioned approaches have models that run somewhere on the cloud and there are privacy and security concerns as resume/CV contain PII (Personally Identifiable Information) data.

Techniques like pruning and quantization are commonly employed to optimize AI systems for resource-limited environments or for EdgeAI [4]. Frameworks like Pytorch Mobile and TensorFlow Lite allow AI applications to run efficiently on devices that come with limited resources, such as embedded systems while maintaining robust performance.

The term EdgeAI is used to describe the deployment on AI algorithms and models on a local edge device such as smartphones, wearable health monitoring devices, IoT sensors etc. One of the popular applications of edge AI is autonomous driving or self-driving car such as Argo AI.

Apple Intelligence is one of the latest examples where we have seen mainstream adoption of EdgeAI that is based on the underlying principle of a personalized assistant that has dedicated hardware on device to run these AI models locally.

2. Technical Documentation

2.1 Application Overview

The Career Coach application was developed to address the challenges faced by recent IT graduates in Oslo during their job search process. Through extensive user research and persona development, the project identified key pain points, including limited work experience, lack of understanding of best practices to get hired, and uncertainty about the specific IT roles to pursue.

The project began with a comprehensive research methodology that combines multiple approaches to understand the needs of recent graduates entering the IT job market in Oslo. The research phase was structured into several key stages:

2.1.1 Initial Research Phase

The initial research phase is the foundation of the project where we were exploring and trying to capture real world challenges, and behaves of the target user which will be discussed in the persona section. Our goal here was simple to understand the nuances of a job seeking experience for recent IT graduates in Oslo. This meant to identify key pain points and validate the potential solutions thereafter. This face combined a variety of methods - qualitative research method, observational studies, and simulated interaction. This helped us in having a strong grounded basis for all our upcoming design decisions.

Understanding the User Context

Our research began with a commitment to truly understand the daily challenges of our target user - "John the Job Seeker", a persona representing recent IT graduates looking for a job in Oslo. Recognizing that the transition from academia to the industry is quite scary and filled with uncertainty, we wanted to capture not only the obvious needs, but also some unspoken subtle frustrations encountered during a job search process.

- *Contextual inquiries* : We conducted several contextual inquiries where we role-played as researcher-recent IT graduate trying to apply for a job. With most of the team acting as potential job seekers in the natural environment, the job was simple but complicated at the same time. We had these sessions every week that involved shadowing people during their job search activities - for instance - while browsing through online job portals are preparing application documents and so on. We noted not only what the users did, but also how they felt at various stages of their journey. An example here would be where a lot of the sessions had people anxious, especially when navigating overly complex job portals. Another common frustration was receiving generic and non-actionable feedback on their resumes.

- *In depth Interviews* : To compliment our observations, we also conducted in-depth interviews with recent IT graduates. These interviews, although informal provided a platform for users to voice their concerns, then expectations and aspirations from a platform that they would like to build if it were up to them. Questions arranged from "What challenges did you face when updating your CV" to "How do you decide which job postings to pursue?". The qualitative data help us identify some recurring problems- such as the need for personalized feedback, clarity in job matching and also enhanced guidance with the interview preparation part. This process help to reinforce our hypothesis that a user centric AI powered career coach could address these challenges effectively.

- *Team Brainstorming and Ideation Sessions* : Parallely, the entire project team participated in brainstorming sessions every week. Initially around 20 team members contributed to diverse ideas and as the project progressed we had a consistent number of 17 team members. In these sessions, we were given a task to present individual ideas and solutions without holding it back and ideas arranged from simple CV passing Singh algorithms to more sophisticated multi modal interaction systems, which could combine all audio, text and visual data.

  The value that these brainstorming sessions provided us was in the diversity of perspectives. We had people from different professional backgrounds - technical experts, user interface designers, data, scientists, and project managers. The democratic process for contribution of ideas enabled every voice to be heard - which Foster is a culture of innovation and curiosity. All these ideas ultimately conversed into a single, cohensive concept: a personalized career coach that could guide the users all the way from trying to find the right job for you till you are preparing for an interview for the job.

Behavioral Mapping and Empirical Validation

A critical aspect of our application was to have a detailed mapping of the entire job application journey. We try to understand the process right from the moment a user becomes aware of a job opportunity to the final decision to apply. This mapping came as a result of both direct observation, and the simulation exercises mentioned above.

- **Mapping the Job Application Journey** : a team was able to create comprehensive journey maps, which captured each phase of the job search process, each one with a unique take on what decisions to make at what time. Caregiver able to identify some key touch points, which included the initial job discovery, CV creation, and customization, application submission and about the follow up process. For instance, during the CV preparation phase uses of struggle with translating their academic achievements into professional skills. This could mean for something like articulating the value of their project or an internship - a gap, if addressed could significantly boost their chances of success. By having a visual map, we were able to find out areas with the most friction. This was shown to highlight the obvious barriers and some subtler ones - such as emotional toll of repeated rejections and the lack of constructive feedback.

- **Empirical Simulations** : to validate our observations be conducted a series of simulations of a job obligation scenario. This involved role-playing exercises where participants went through the entire application process with both the traditional method of doing it and a preliminary prototype of our proposed system. The simulations allowed us to test the efficacy of our interventions - such as the automated CV feedback and real time job matching recommendations. Here we also measured some key performance indicators - one of which is the time taken for coherent, applicable feedback; the accuracy of job matching among others. The one key observation was that users are more relieved and with increased confidence once they felt they had received clear and actionable insights may be CV improvement or either matching them to a well matched job position.

- **Iterative Validation through Prototyping** : A key component of a research was the development of low fidelity prototypes. Early versions were created to test individual features such as CV representation and analysis in chat. These prototypes were iteratively improved where participants interacted with the system and provided real time feedback. For example, in one iteration, a prototype which provided detailed information on the homepage related to a job description was discarded in place offer metric based description. This was done to maintain the minimalistic outlook of the application, whereas we would not want the user to have cognitive overload in the first glance. This iterative loop insured that our design was continuously, refined and grounded in both user needs and developer requirements.

Pain Point Documentation & Synthesis

The research face helped us come up with a detailed synthesis of all the collected data. We compiled all the findings from interviews, the observation studies, and the exercises into one comprehensive report. Now this report documents, the explicit pain points, and also the more nuanced latent needs of the target audience.

The key pain points identified are as follows:

1. **Limited, and Generic feedback** : As many users claimed, the existing systems were only providing superficial insights into their CVs. This was a terrible approach, considering each candidate's profile has unique strengths and weaknesses.

2. **Uncertainity in Job Matching** : There was a lot of confusion about how to navigate the vast array of job listings, the lack of personalis matching would lead to a lot of waste of time and missed opportunities.

3. **Emotional Barriers and Lack of Confidence** : repeated failures, and rejections, along with the lack of constructive feedback, led to a sharp decline in user confidence with time. The emotional effects of the job service process lead to users, feeling overwhelmed and uncertain about their professional prospects.

4. **Technical Hurdles in CV Customization** : Beyond the general feedback, there were some specific challenges related to formatting, understanding the context of the candidate profile, and the translation of their academic or project based experiences into a more professional setting

In conclusion, these findings lead to several important design, implications. Firstly, the system should be required to offer detailed and context of your feedback to the user rather than a simple fail or pass evaluation since that would lead to further confusion. Secondly, the job matching component had to become capable of understanding and adapting to the diverse profiles of its users. Thus having both qualities of being dynamic and being personalized. Thirdly, the overall user experience had to be designed so that we are able to reinforce user confidence in the system with supportive and real time interaction. Finally, the system also needed to handle diverse range of resumes.

Reflection on the Research Journey

This initial research phase was just not a preliminary step, but also an immersive journey into our users world's. My understanding and observing their interactions deeply with the existing systems, regained a holistic understanding of the different challenges that they faced. This deep dive provided the found foundation which was necessary to design a system, which is not only technically robust, but also very empathetic in nature to its users needs.

The insights derived here helped to make every subsequent design decision later. For example, the clear need for a personalized and context, sensitive feedback influenced us to integrate a Retrieval Augmented Generation (RAG) architecture in our system. Similarly for removing the cognitive overload, and also building user confidence in the system - led to incorporating a responsive chat interface and also a instant company logo detector using YoloV8.

In conclusion, this phase was critical in transforming abstract ideas into a concrete user centered design vision. By combining empirical methods with an empathetic view of engagement, we henceforth let the groundwork for a system, which not only needs the technical requirement, but also addresses the emotional and practical challenges, faced by the user. We leave the section now to proceed with all the inspiration practical insights gained to understand how this helped validate us at every design choice based on real-world data, and user experience.

2.1.2 Persona Development

The development of "John the Job Seeker" persona, shown in Fig.2.1, emerged as a crucial foundation for the Career Coach application, representing the synthesis of extensive user research in Oslo's IT sector. This persona embodies the typical challenges and aspirations of recent IT graduates, providing a concrete framework for technical implementation decisions.

The primary goals of John:

1. **Gaining Practical Job Analysis:** This directly influence the system's core architecture. The integration with Arbeidsplassen API wasn't just a technical choice - it represents a deliberate effort to surface entry-level positions suitable for new graduates. The system's intelligent filtering mechanisms, powered by the Qwen-1.5B model, were specifically tuned to understand and evaluate early-career qualifications, evaluate benefits, working arrangements within job descriptions. This analysis extends beyond simple keyword matching - the system understands contextual clues about company culture and work environment, particularly valuable for recent graduates seeking their first permanent position.

2. **Accessing Networking oppurtunities:** The networking aspirations of our per-

sona led to the one of the most technical implementations: the logo detection system. Using YOLOv8, we aim to transform physical company encounters into immediate job opportunities. This feature bridges the gap between physical networking and digital job posting, particularly valuable for those like John who are building their professional network from scratch.

3. **CV customization:** This shapes the next crucial layer of implementation where the system employs a specialized customization for CV. The need for CV analysis into achieving certain objectives like project-focused CV templates, skill extractions, integration of academic and personal project descriptions allows for creation a better personal profile for becoming a better candidate in a particular job context. This is acheived finally by the LLaMA3.2:1B's implementation along with RAG.

The pain points of John are:

1. **Limited feeback on job applications:** The lack of feedback on his applications has been particularly frustrating for John. Without understanding why his applications aren't successful, he's unable to improve his approach effectively. The system's real-time CV analysis provides him with specific, actionable feedback - for example, suggesting ways to better highlight his programming projects or identifying key technical skills he should emphasize based on current job requirements. This continuous feedback loop helps John refine his applications iteratively, learning from each submission.

2. **Uncertainity about IT roles:** This makes it difficult to target his job search effectively. The system helps him navigate this complexity by mapping his existing skills against various IT positions, showing him where his academic background aligns with market demands. Through integration with Arbeidsplassen data, John can see which roles are actively hiring entry-level positions and understand the specific requirements for each path, helping him make informed decisions about his career direction.

3. **Lack of professional network:** John's lack of professional network compounds his experience gap. Being new to Oslo's IT sector, he has few industry connections and limited knowledge of potential employers. The logo detection system transforms his daily encounters with company logos into networking opportunities. When John spots an IT company's logo during his commute or while exploring Oslo, the YOLOv8n-powered system instantly provides him with company information and relevant job openings, turning casual observations into potential career opportunities.

4. **Limited work experience:** As a recent IT graduate in Oslo, John faces several interconnected challenges that make his job search particularly daunting. His limited work experience - primarily consisting of academic projects and perhaps a short internship - creates significant anxiety when applying to positions that seem to demand years of professional experience. The system addresses this core challenge through context-aware CV analysis, helping John translate his academic achievements into professional value. For instance, when John inputs his final year project into the system, the RAG architecture analyzes it to highlight professional-level skills he might not have recognized - like agile development practices or system

architecture design - and suggests ways to present these effectively to potential employers.

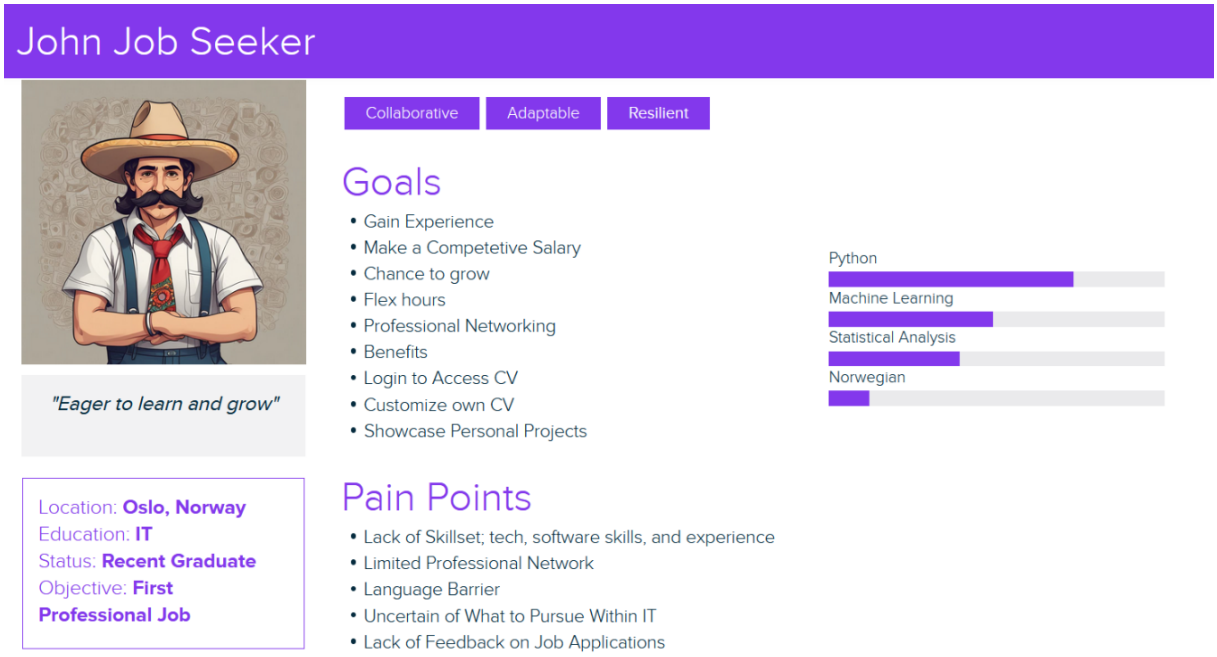


Figure 2.1: John the Job Seeker's Persona

2.1.3 Design Decision Framework

Are research highlighted that recent ID graduates - shown by our persona "John the Job Seeker" . Faces certain challenges, such as limited actionable feedback, uncertainty while navigating job listings and general lack of confidence arising because of impersonal and generic systems. To answer most of these problems are framework is constructed in such a way that it not only empower the users by offering them a tailored experience and guidance, but also robust real time feedback, this helps for a continuously improving user experience.

Core Principles Derived from Research

- **User empowerment** : The above said research clarified a few things, that uses require more than just a technical tool - they need a system that can both educate and be confidence in them at the same time. And our framework is rooted in this idea. Every Design decision made is with a goal to move from abstract data to a personalized, actionable feedback system, which not only empower the user to refine the CVs, but also make informed career decisions.

- **Personalised, Context Aware Feedback** : Traditional systems usually provide either binary or generate evaluations. We used this to our advantage while using Retrieval-Augmented Generation (RAG) techniques to help deliver a detail and contextual feedback to the user. When a user submits a CV, the system segments the document into chunks, converts those chunks into a vector representation, and

then retrieves only the most relevant information. This process helps the system make potential improvements on the fly and also helps answer key questions such as how highlighting, a particular skill or restructuring, a project description could enhance the overall profile.

- **Rapid, Localized Inference** : since speed and responsiveness are key to building trust we designed a system with local deployment of efficient and compact models (such as Llama, TinyLlama and Qwen) in mind. This local set up, helped ensure that we were able to deliver responses in real time. Since maintaining a fluid conversation helps reinforce use confidence.

Implementing the Framework

1. **Feature Confidence Building**
   Based on our research, one of the critical challenges is the lack of confidence arriving from vague, or overly technical feedback. So to counter this, the CV analysis component of our system employees the RAG technique of ingesting information. By dynamically in integration, the most relevant context the system now provides feedback that is most relevant as well. For example, if John's CV mentions "project management" without any other detail, the system might suggest rephrasing it to include specifics such as "coordinated cross functional teams, using agile methodologies". This level of feedback makes the recommendations, actionable and demystifies the improvement process.

2. **Establishing Robust Feedback Loops**
   Our framework emphasizes on the importance of an iterative interaction. We do that with two key mechanisms in place.

   Firstly, the system incorporates features like real time company logo detection using Yolo V8. Say when John scans the logo anywhere in the street, the system instantly identifies the companies logo and pulls up relevant job listings. Simultaneously the real time chart interface - which is powered by a locally deployed, LLMs will be able to answer to any questions he might have.

   And secondly, every time John updates his CV based on the systems recommendation, a reanalysis process is taken in effect. This cyclic process allows the system to find tune it suggestions based on the ever evolving contents of the CV. Simultaneously the job matching algorithms will also be refining their recommendations based on the users interactions. This ensures that the learning happens on both sides, the user as well as the system.

3. **Balancing Technical and User-Centric Requirements**

   A significant effort went into, ensuring that the system remains robust, despite the hardware contains. That means to enable efficient processing of resource demanding tasks, such as vector and bedding rack, processing and working of the LLMS, and other system frameworks across the resource constrained Raspberry Pi cluster. Thus a distributed approach for competition needs was fundamental to maintain the responses of the system, and to ensure the system remains scalable with time.

Here, we were able to bridge user challenges and strategic development strategies. These design decisions mentioned above were not made in isolation. They provide for a solid foundation that directly helps our feature prioritization process.

2.1.4 Feature Prioritization

The team based on user impact and technical feasibility proceeded forward with the following categorization:

Must-Have Features

1. CV Analysis and Enhancement (Priority: High — User Impact: Critical) The CV analysis system emerged as the cornerstone feature, implemented through RAG(Retrieval Augmented Generation) architecture for contextual analysis. The technical implementation of this feature focused on:
   - Skill extraction from CV
   - Real-time feedback generation
   - Integration with job description matching
2. Job Matching System (Priority: High — User Impact: High) The job matching system allows for most relevant jobs first with real-time job listings coming in from Arbeidsplassen API. Company recognition thorugh YOLOv8 logo detection. The technical implementation of which focuses on:
   - Filtering IT-specific positions in Oslo
   - Providing contextual job recommendations
   - Integration with job description analysis
3. Interview Prepper (Priority: Medium — User Impact: High) The interview prepper covers for most of what is left after the above two features. Finishing the last stage of preperation. It was implemented with a few resource constraints (resource and time) in mind. The technical implementation of which focuses on:
   - Focus on IT-specific interview scenarios
   - Achieving real-time response times and conversational capabilities.

Could-Have Features

1. Network Building (Priority: Medium — User Impact: Medium) This feature was designed but with limited scope:
   - Company recognition through logo detection
   - Integration with job listing data
   - Basic professional connection tracking
2. Industry Insights (Priority: Low — User Impact: Medium) While valuable, this feature was de-prioritized due to:
   - Resource constraints on edge devices
   - Focus on core job-seeking functionality
   - Complexity of real-time market analysis

The team's decision to prioritize CV analysis and job matching proved particularly important given the importance to resource constraints, where success of core features depends on careful optimization and efficient model selection.

2.1.5 Key User Requirements

The primary target user, represented by the persona "John the Job Seeker," is a recent IT graduate in Oslo seeking their first professional role.

The identified key user requirements include the following:

- Need for practical job experience and networking opportunities
- Desire for stable employment with benefits and flexibility
- Assistance with CV customization and project portfolio presentation
- Guidance in navigating the local IT job market

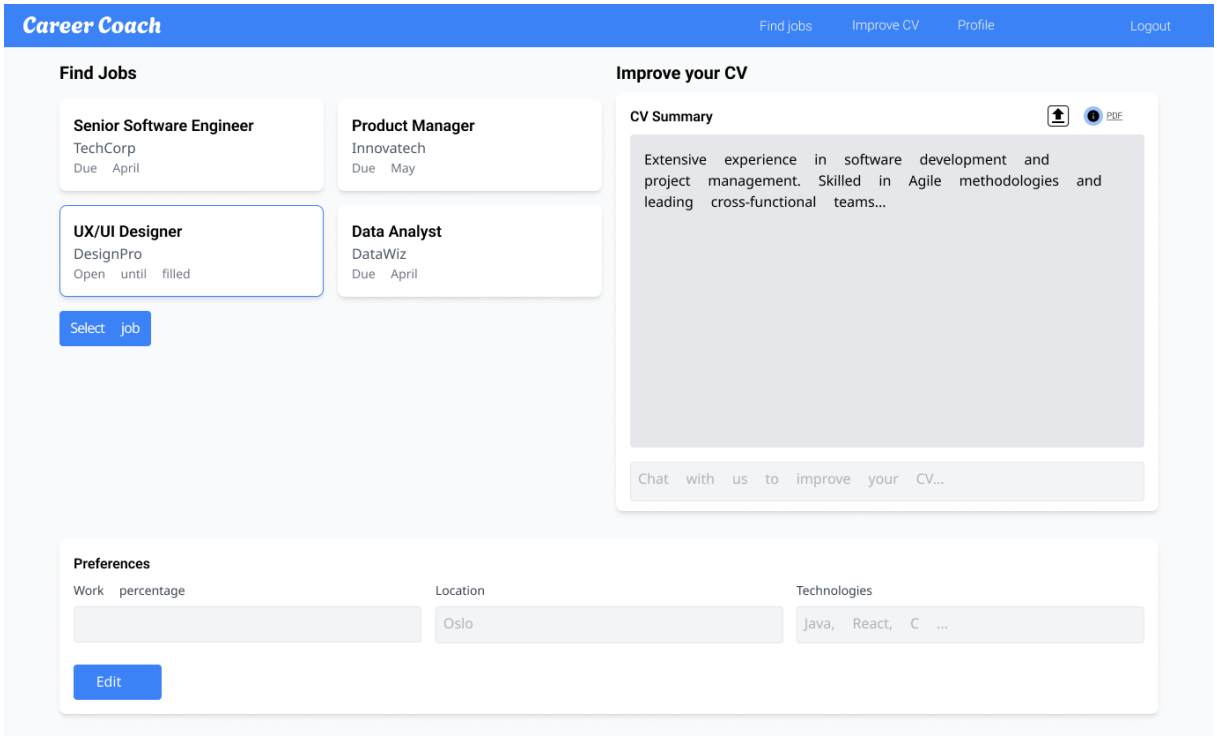


Figure 2.2: Screenshot of the overall Career Coach interface

2.1.6 UI Design

The primary goal of user interface is to create a clear and supportive environment with a user is able to effortlessly explore opportunities, refined their CV and prepare for the interview interviews thereafter - receiving continuous and protection feedback continuously. We designed the layout navigation and the interactive elements around three core UX principles, **transparency, guidance, and consistency.**

Layout and Navigation Flow :

At the top of the interface is a persistent navigation bar which features the career coach brand identity along with section links for: *Find Jobs, Improve CV and Profile.*

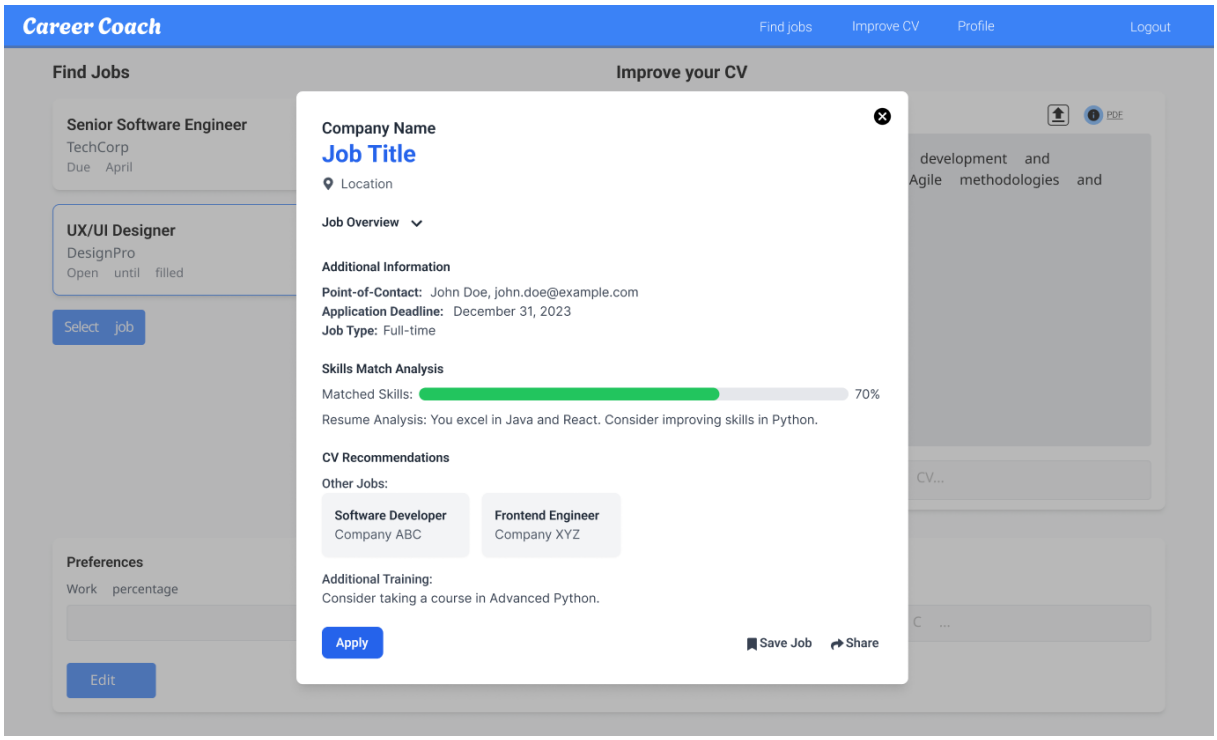


Figure 2.3: Screenshot highlighting the dashboard and navigation elements

This minimal structure and shows that the user is not overwhelmed with options and can switch tasks without much confusion.

Below is the navigation bar where the screen splits into 3 different functional panels - each associated with a specific user task. For instance, when selecting *Find Jobs*, the left panel would display a scrollable list of relevant positions - in descending order of matched opportunities based on your resume. It would only include concise labels like the company, name role, title, and due date, all of which essentially to making a decision to click on it or not. Once a user clicks on a listing, pop-up appears with a more detail details, job overview, including a skill, match analysis and also a quick application button. This modular approach here, prevents cognitive overload, and allows for users to only consume the information that they need at that step.

Improved CV Panel :

When switching to the improved *CV section*, users can see a summary of the current resume, such as the extracted skills and experience highlights. Progress work for skill based matching communicates, visually how closely a severe aligns with the selected job requirements. Thereafter system generated recommendations generated below that will offer actionable insights. For example, suggesting to add some technical skills to highlight or ways to rephrase a certain project experience. This part is where the underlying AI system (RAG-augmented architecture) truly shines.

Interaction patterns are kept simple such as the interaction to upload the newer and updated résumé is a single upload button near the CV summary header. This would prompt the system for an immediate analysis. The interface also uses colour coated highlights to show what has changed guiding users towards a deeper understanding of how each edit is impacting their overall matching score.

Interview Prepper :

This module presents with a chat like Q&A format. While at the top a compact card displays the target job description, and some key resume highlights. This sets the stage for the context where interview simulations. With each system generated question appearing in a conversation bubble the user can choose to respond with a response. At the end of the interview, a collapsible feedback panel appears, this would offer for structured observations, for example, "strong technical depth", or "lacks clear impact matrics" and some improvement tips. Since this chat, based layout feels more natural replicating a back-and-forth in a real interview scenario, but also encouraging iterative refinement of answers on the go.

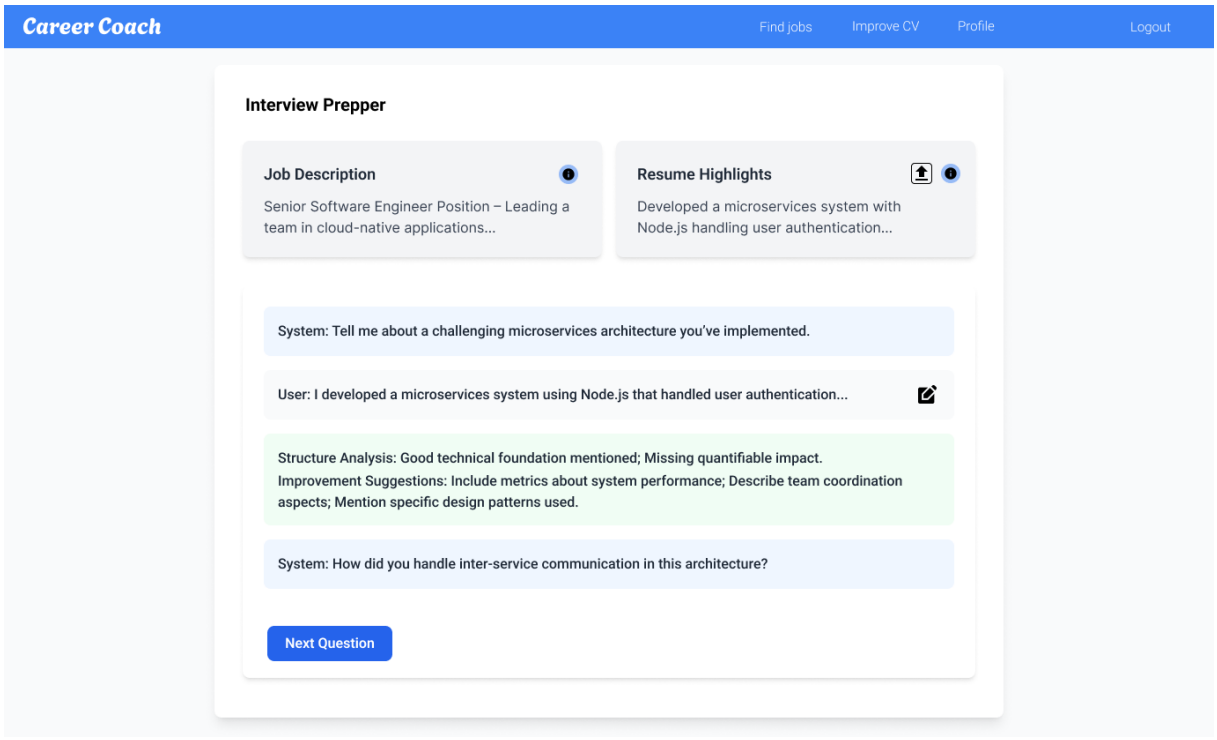


Figure 2.4: Screenshot highlighting the interactive chat interface and interview preperation section.

To maintain momentum and engagement with the system, *Next Question* button immediately prompts up once one question is effectively answered. This platform allows for the user to experience and practice for multiple types of questions which could be technical behavioral or cultural. And by keeping the conversational interface, compact and easy to read users can quickly review past their QA pairs tracking their progress overtime.

Visual Consistency and Branding :

Across all sections, we insured to have a consistent colour scheme and typography that reinforces a single cohesive brand identity. High contrast, text and intuitive icons, ensure readily throughout the interface while neutral backgrounds. Keep the focus on actionable content. This was also done intentionally to keep the UI minimalistic in nature, which makes it much easier to both develop and use. Importantly everything was optimized for performance, ensuring it would not disturb a users workflow on the low power devices like

Raspberry Pi's.

Accessibility and Responsiveness : The interface is built to be responsive so that resizing from desktop to tablet to mobile views is graceful. Key interactive elements, such as the buttons from fields and navigation links have been sufficiently spaced for a good user experience, both on desktop and touchscreen device devices. Accessibility best practices, for an HTML structure and high contrast, modes and scalable text allows for accommodating, diverse users needs.

Iterative UX Testing and Feedback :

Throughout the development, user testing sessions were conducted with prototypes to validate navigation clarity to discover some points of confusion and friction. Observation/-confusion around skill, matching scores led to a *colour coded scheme* for understanding it well. Another observation upon when does a user know a particular question is done being answered was solved by the *Next Question* button.

This user centric and iterative design approach and ensure that every UI element - from a job listing card to an interview. Question bubble served a purpose and feels intuitive to use for the target audience. The result is a concise and coherent interface simplifying the jobs for journey, offering immediate insights guidance and encouragement, every step of the way. Thus by aligning these design choices with our AI capabilities, we were able to deliver a cohesive experience, which empower is the user to take confidence to navigate the competitive landscape of IT job hunting.

2.1.7 Literature Survey on AI research

The evolution of natural language processing(NLP) techniques over a decade has lead to significant advancements in automated document analysis, one use case for which is Resume (CV) Analysis. Automated resume analysis or screening was predominantly a rule-based system and a keyword matching technique based system. These methods, were effective in reducing the human effort required for analysing larger amounts of resumes, but they were limited in their abilities for capturing contextual nuances and contextual semantics. And as the volume and diversity of resumes increased academic and industry, there came a need for research in more advanced machine learning techniques - in this case, particularly transformer based models.

Early Approaches and Limitations

Early on, these resume analysis or screening systems were heavily dependent on predefinition of rules and simple statistical models. They primarily used keyword extraction to match candidate skills with job descriptions. Even though now the systems were able to reduce the time required for the initial analysis, they often fail to recognizing these nuances in how semantically similar skills would be described for example one resume might list project management while other might say, coordinating cross functional teams, both of them convey similar capabilities. As noted by earlier studies [5], now these keyword based approach was pronto a lot of errors due to their inability to capture context and understand nuances in the language so this drawback by the of the rule base systems motivated research into exploring more sophisticated methods, which would bridge this gap.

Some early work into this research [5] involved the use of conventional machine and techniques such as support vector machines or SVM and decision tree. These help to classify resumes into a predefined set of categories, but were limited by the need for extensive feature engineering and also their inability to be able to generalize well across structured and diverse ranges of resumes. The unexpected resume layouts of styles and different ways of expressions compounded the challenges thus resulting in systems which were better but error-prone.

Transformer Architectures: A New Paradigm

The introduction of the transformer model (Vaswani et al. 2017) [6] marks a huge advancement in NLP with its self attention mechanisms enabling models now capture long range dependencies in text, unlike its producers like SVM, which were prone to not being able to understand context in the long-term. Now this particular ability was beneficial for processing resumes, which often required to understand. The context spread over some sections, and these transformer based models demonstrated substantial improvement in a variety of language realated tasks, and their potential for resume analysis has been explored extensively since it's release.

One of the earliest studies applying these contest, aware, transforming models for understanding the competency level, and also matching resume a job description was presented by Li et al. (2020) [7]. In their work, they are annotated a large data set of resumes and used these transformer models to classify resumes into different experience levels. Their approach highlighted that even small improvements in understanding semantics achieve through this transformer based and coding could lead to significant gains in tasks by leveraging their multihead attention, mechanisms and section specific encoding strategies. Their model was able to achieve accuracy of up to 73.3% for classification tasks showing that the potential of transformers to enhance this automated resume screening.

This study along with many others build the foundation for exploring fine tuning strategies, adapting pre-train transformer models to a specific domain such as human resource management and models such as Bert Roberta and they are variance showed that they could extract in detail information from the resumes ranging from educational qualifications, work experiences to soft skills and personality traits [8]. These models were able to not only capture just the text itself, but also the context thus addressing one of the primary shortcomings of the earliest systems as discussed.

Retrieval-Augmented Generation (RAG)

Now, given an understanding that transformers were able to capture contextual information. One challenge remain persistently now since transformer models by design, have a fixed context length, and rely only on their internal static training data for their reasoning capabilities. This limitation would lead to an outdate or incomplete information in a sense where now the large language model would rather hallucinate than provide you the correct information. This is particularly evident in the case of processing resumes because that could contain rapidly evolving details that might not have been there while the model was being trained retrieval, augmented generation or ragged techniques have emerged as a promising solution to this problem.

RAG integrates an external retrieval mechanism with the generative capabilities of a transformer model. No, instead of only relying on the model speed train parameters

allows for dynamically, retrieving relevant texts from an external large corpus of information there by extending the models effective context. As demonstrated by Lewis et al. (2020) [9], racked significantly improved performance on knowledge intensive tasks by incorporating up-to-date information from an external data source, and in this context of analysis this meant that instead of processing an entire lengthy resume often filled up with filler words and too much detail the system now only retrieves the most relevant chunks. Example work experience in skill to inform its analysis.

A typical RAG process involves several key steps:

- **Dynamic Chunking** : Resumes are divided into smaller overlapping segments to ensure that the important context is preserved. This helps not only to reduce memory overhead, but also helps in more precise retrieval
- **Embedding and Indexing** : With each chunk now transformed into a dense vector representation using a state of the art abiding technique. These embedding are then stored into a vector database for effective similarity search.
- **Retrieval** : On receiving a query, safe for instance, identify key technical skills in this resume for AI engineer role, the system now includes the query into a embedding and perform the similarity search over the already index chunks, which are also in in embedding form. The top relevant segments are retrieved.
- **Generation** : now the dog, relevant segments retrieved are integrated into the prompt for the large language model, which now generates more context of their response that reflects to its knowledge received, and also about the candidate skills relevant to the AI role also present now in the embedding.

The dynamic nature of rag allows for real time update and also improves the systems robustness against its limitations of a static training, data, and hallucinations that come along both of which are one of the biggest challenges in large language models - also a transformer based architecture - even now! Another approach called RAGGED (2024) [10] found this approach, by improving the quality of this retrieval, and also the method by which the retrieved information is integrated. This greatly affects the overall performance of the system, and then the insights are crucial for designing a resume and system that can deliver both high accuracy and efficiency.

Fine-Tuning Techniques for Domain-Specific Applications

The evolution from large pre-trained language models which are trained on large corpora of text, having very robust general language understanding capabilities. But their performance on some domain specific tasks - such as resume analysis - requires for further adaptation and improvement. This adaptation can be received through a process called as fine-tuning where these general models are now specialized for a specific application by training them on relevant data. In our context, the fine tune model is now capable of understanding, the subtle nuances in resumes such as weighted, formatting, domain specific terminology, and also different expressions of work experience and skills.

Several methods have been researched, which can efficiently find these large language models under some resource constraint. This resource constant is different from the compute restrictions from the Raspberry Pi cluster, but rather general access to high-end

computer devices. Techniques like Low-Rank Adaptation (LoRA), and it's efficient variant, QLoRA, enable fine-tuning, a large language model by adjusting only a small subset of all the parameters available. This enables for drastically, reducing both competition and memory overhead required during the fine-tuning process. Research by Dettmers et al. (2023) [11], demonstrated that QLoRA help and efficient adapting of a quantized large language model without sacrificing performance - so this capability help us to find our large language models on a local machines with a decent GPU. Another research by Hu et al. (2022) [12] found that LoRA is an effective approach for fun evening a large language model with quite minimal additional training overhead.

The need for fine-tuning became relevant with the results of our early experiments. Since most large language models with a smaller number of parameters around 1 billion and a smaller context window, were not able to consistently reason and provide feedback. We proceeded with fine-tuning. So even though Gemma:2B and TinyLlama:1.1B provided for a decent baseline they were not able to capture the complex and context which nature of resume data. Fine-tuning on a custom data set seemed like the next best thing to do. The custom data set was generated using a combination of PDF text extraction, basic normalization, and some domain specific feature extraction namely Named Entity Recognition (NER). This extracted data was the used to generate a set of questionnaires and answers generated by another large language model - Mistral:7B [13].This structured way of presenting the data provided for the model to understand the very behavior in which a question has to be answered in our specific context. Multifaceted prompting during fine tuning help by dividing the task into sub tasks like general feedback, skill, extraction, and alignment evaluation - disprove to be effective strategy for our domain specific use-case. The idea that context aware transformer models are significantly benefited from task specific prompts which results not only in improved semantics, but also better actionable insights for the user was proved by a research (Li et al. (2020) [7].

The integration of fine-tuning techniques improve the overall accuracy. By optimizing the hyper parameters (for example, using gradient check-pointing, FP16 mixed precision training, and gradient accumulation), we tried to ensure that the fine-tuning process was not only efficient, but also scalable in the long term. Thus fine tuning emerged as a critical step in bridging a gap between a general purpose model and a domain specific resume analysis system. But in theory also should have enhanced the moral's ability to operate effectively within the enhanced use of its constrained context window and our constrained resource limitation.

Research Gaps and Future Directions

The literature provides for a strong support for use of transformers and drag in automated analysis, but several research gaps still are remaining. First of which is the optimal method for dynamic chunky based on various sections of the resume. Since the different sections of a resume, such as work experience, education and skills may require for different chunk sizes to capture all relevant details efficiently. While fixed length chunking technique with overlaps is a common strategy. Adaptive chunking allows for adjusting to a newer document structure, which may have yielded into a further improvement in performance.

The second gap remains even while RAG techniques and fine-tuning techniques have been successfully applied in knowledge intensive domains their integration into this one new case of resume is still quite an uncharted territory. This challenge also came into effect while we were studying for different data sets for fine-tuning approaches for a large language model the majority of existing rag research focuses on a very general NLP task

or specific means such as healthcare. While rack is one of the ways to achieve this solution of going from static databases to dynamic one tool calling for a search tool can collect relevant information on the spot, could be a means for newer and efficient way to have an updated knowledge base.

Finally, scalability that means a very significant challenge. Research into different and better ways to either fine tune, a smaller model for this task or distillation of a bigger model, which is capable in resume analysis into a smaller model can help. A few attempts were made to somewhat successful. Most of it was not relevant enough in its answers to explain and analyze resumes due to a smaller context window for the models experimented with at time.

As the field continues to evolve, future research must focus on adaptive chunking strategies, domain-specific optimizations for RAG. The integration of these advanced techniques promises to transform the automated resume analysis into a powerful tool, enhancing the efficiency of HR processes. Concluding the research from Li et al.(2020) [7], Lewis et al. (2020) [14], RAGGED (2024) [10], Resume2Vec (2024) [15] and ResumAI (2023) [16], this work in itself is the intersection of the state-of-the-art transformer models, retrieval-augmented generation, and distributed system design at the time of writing. It builds upon the solid foundation of previous research while addressing critical gaps from before.

2.1.8 Architecture

Architectural Evolution:

The journey of developing our career coach application's architecture proved to be an exercise in iterative refinement. Like many ambitious projects, we began with what seemed like a straightforward approach, only to discover the complexities that would shape our final implementation.

Our initial architecture envisioned a direct connection between front-end and AI models - a design that appeared elegant in its simplicity. The concept was straightforward:

Initial Design Components:

- React front-end making direct API calls to AI models
- Simple database structure for user data
- Basic authentication layer for security
- Direct model inference for user requests

However, as we started implementing this design, we encountered several critical limitations that would force us to fundamentally rethink our approach. The challenges manifested in three key areas:

1. **Performance Management:** The direct communication model quickly revealed its limitations:
    - Concurrent user requests created unexpected bottlenecks
    - AI model response time became unpredictable
    - Resource allocation across different request types(CV analysis, job matching, logo detection) proved inefficient

- Memory management on our Raspberry pi nodes showed concerning patterns

What particularly struck us was how these issues compounded under load. A simple CV analysis request could suddenly take significantly longer when multiple users were accessing the system simultaneously.

2. **Data Flow Complexities:** Our initial data architecture as shown in Fig.2.5 proved to be particularly problematic, with issues such as:
   - Inconsistent data states between sessions
   - Difficulty maintaining data integrity
   - Growing complexity in handling different data types
   - Inefficient query patterns impacting response times

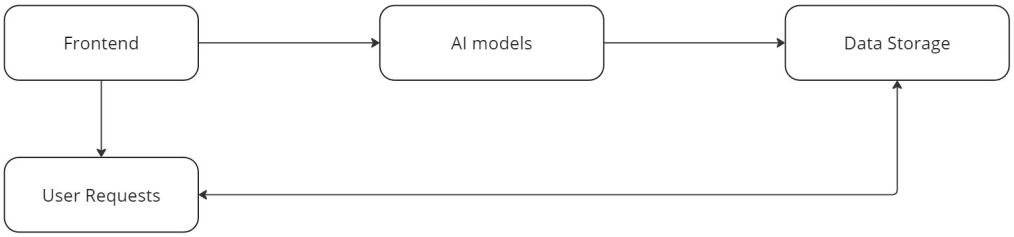


Figure 2.5: Initial Architecture Diagram

The breaking point came when we realized that scaling this architecture would require exponentially more resources for each additional user. This realization led us to our revised architecture - a more robust and scalable solution that would better serve our needs.

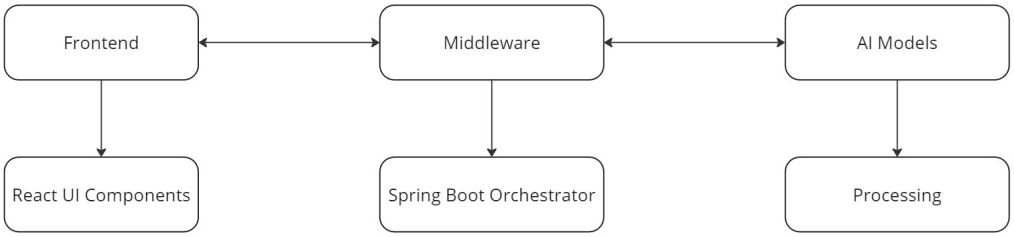


Figure 2.6: Revised Architecture Diagram

The new architecture, as shown in Fig.2.6 introduced the Core Integration Layer because of which we could achieve the following:

- Centralized middleware service for request handling

- Intelligent request routing and load balancing
- Sophisticated session management
- Robust error handling and recovery
- Efficient cache management

The effectiveness of this revised architecture became immediately apparent. Response times stabilized, resource utilization became more predictable, and most importantly, we gained the ability to scale individual components independently based on demand.

Component Architecture:

The evolution of our initial design led us to a highly modular component architecture, where each element serves a specific purpose while maintaining clear integration boundaries. This approach proved essential for deploying sophisticated AI capabilities on our Raspberry Pi infrastructure.

1. **Frontend Implementation:** Our front-end architecture, built using React, focuses on delivering a seamless user experience while efficiently communicating with our middleware layer. The implementation follows a careful balance of functionality and performance:

   User Interface Components:

   - Dashboard for job listings and recommendations
   - CV upload and analysis interface
   - Real-time chat functionality
   - Company logo detection integration

   We encountered interesting challenges in implementing the chat interface, particularly in maintaining responsive interactions while processing AI responses. Our solution leverages Llama3.2's capabilities through an optimized interface:

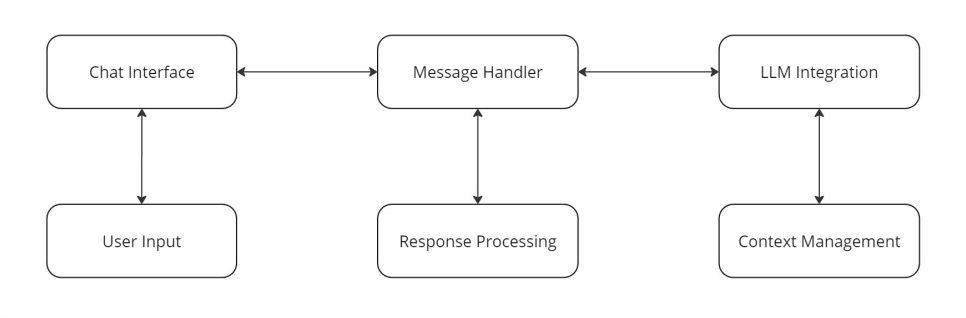


Figure 2.7: Frontend System Flow Chart

2. **Middleware Layer:** The middleware service emerged as the cornerstone of our architecture, handling the critical tasks of authentication, request routing, and data management. Built using Spring Boot, this layer orchestrates all communications between the front-end and our distributed AI models.

   Key Middleware Functions:

   - Session management and authentication
   - Request routing to appropriate AI models
   - Database transaction coordination
   - Real-time job listing updates via Arbeidsplassen API

   One of our most significant technical achievements was implementing efficient request routing:

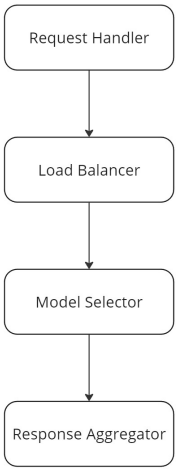


Figure 2.8: Middleware Flow Chart

3. **AI Model Distribution:** The distribution of our AI models across Raspberry Pi nodes required careful consideration of resource constraints and processing requirements. We implemented different agents for defining our multi-modal agents satisfying our criteria of the project for both resource limitations and multi-modal agents implementation.

   Processing agents:

   - CV Analysis agent using RAG architecture
   - Job and user matching
   - Interview Preparer for interview related question-answer
   - Company identification and future connections

The integration between these components reflects our commitment to efficient resource utilization while maintaining system reliability. Each component operates within clearly defined boundaries yet works seamlessly as part of the larger system.

Database Architecture:

The complexity of our application, aims to handle everything from user profiles to AI-generated insights, demanded a sophisticated approach to data management. Our experience led us to implement a dual-database architecture, each optimized for specific types of data and access patterns.

1. **Data Management Strategy:** Our initial attempts to use a single database quickly revealed limitations in handling both structured user data and the more fluid, unstructured AI outputs. This led us to develop a more nuanced approach:

2. **MySQL Implementation:** For user-related data and authentication, we leveraged MySQL's robust relational capabilities:

   - User Profiles and authentication details
   - Job preferences and search history

- Session management data

3. **Data Flow Management:** The interaction between our databases follows carefully designed patterns to maintain consistency and performance Transactional Data Handling:

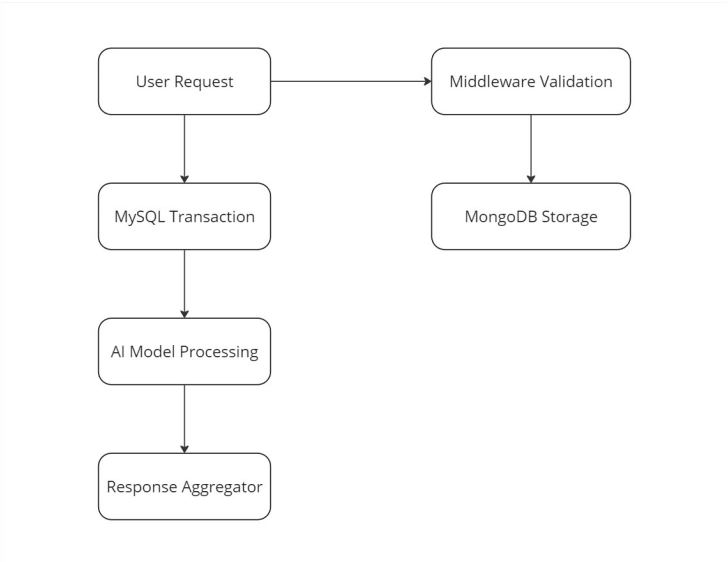


Figure 2.9: Database Flow Chart

One of our more interesting technical challenges involved maintaining consistency between the two databases during complex operations like CV analysis. We propose a solution through a transaction coordinator in our middleware:

Transaction Coordinator: Step 1: Begin MySQL transaction (user session) → Step 2: Process AI model request → Step 3: Store results in MongoDB → Step 4: Update user history in MySQL → Step 5: Commit transaction

This approach ensures data integrity while maintaining the performance benefits of our dual-database architecture. The system gracefully handles common scenarios like:

- Interrupted CV analysis sessions
- Partial job application submissions
- Chat context preservation
- Multi-step interview preparation processes

Through careful optimization and monitoring, we've achieved response times averaging under 100ms for most database operations, with more complex AI-driven queries typically completing within 800ms. These metrics have proven crucial for maintaining a responsive user experience despite our edge deployment constraints.

Logo Detection:
Logo detection is a subfield of object recognition that focuses on identifying and classifying brand logos in images or videos. It is widely used in areas such as brand monitoring, counterfeit detection, and visual search applications. Various techniques have been developed for logo detection, ranging from traditional image processing methods to deep learning-based approaches.

1. **Traditional Approaches:**
   Earlier methods for logo detection relied on feature-based techniques such as Scale-Invariant Feature Transform (SIFT) [17] and Speeded-Up Robust Features (SURF) [18]. These methods extract key visual features from images and use descriptors to match logos across different scales and orientations. While effective in controlled conditions, these approaches struggle with occlusions, complex backgrounds, and variations in lighting.

2. **Deep Learning-Based Approaches:**
   The introduction of deep learning, particularly convolutional neural networks (CNNs), has significantly improved logo detection performance. Region-based CNNs (R-CNNs) [19], Fast R-CNN [20], and Faster R-CNN [21] introduced end-to-end trainable models that allow for better object localization and classification. However, these models are computationally expensive and require significant processing power. Single-shot detectors such as You Only Look Once (YOLO) [22] and Single Shot MultiBox Detector (SSD) [23] provide an alternative by performing object detection in a single pass, making them more efficient for real-time applications. YOLO models, in particular, have been widely used in logo detection due to their speed and accuracy in detecting objects at multiple scales. More recent iterations, such as YOLOv4 [24] and YOLOv8 [25], have introduced improvements in network architecture, loss functions, and feature extraction, leading to enhanced detection performance.

YOLOv4 Architecture:

YOLOv4 consists of three primary components: the backbone, neck, and head, each designed to improve specific aspects of the object detection pipeline.

1. **Backbone:**
   The backbone is responsible for extracting essential features from the input image. In YOLOv4, the CSPDarknet53 backbone is used, which is a variation of the Darknet architecture. CSPDarknet53 leverages Cross-Stage Partial (CSP) connections to improve gradient flow, reduce computational complexity, and enhance learning capacity. It processes images through multiple convolutional layers and feature extraction blocks, enabling robust detection of objects of varying sizes.

2. **Neck:**
   The neck bridges the backbone and the detection head, providing an intermediate feature representation. YOLOv4 incorporates the Path Aggregation Network (PANet) in its neck design. PANet aggregates multi-scale feature maps, ensuring the effective propagation of features from lower and higher levels of the network. This multi-scale approach enables the detection of both small and large logo objects with high accuracy.

3. **Head:**
   The head of YOLOv4 predicts bounding boxes, object classes, and confidence scores for detected objects. Using anchor boxes, the model refines predictions to align closely with the ground truth. The head employs the YOLO detection paradigm, which divides the input image into a grid and assigns predictions to specific grid cells. Each cell predicts multiple bounding boxes with associated confidence and class probabilities.

The Key Features of YOLOv4:

- **Mish Activation Function:** Used in the backbone for non-linear transformations, Mish enhances gradient flow compared to traditional activation functions like ReLU.
- **CSP Connections:** The Cross-Stage Partial connections reduce redundancy and memory consumption while preserving accuracy.
- **Spatial Attention Module (SAM):** This module enhances the model's ability to focus on critical regions of the input image, improving object localization.
- **Bag of Freebies (BoF) and Bag of Specials (BoS):** YOLOv4 incorporates several "free" augmentations, such as Mosaic data augmentation, CutMix, and label smoothing, to enhance training data diversity. Specialized techniques like CIoU (Complete Intersection over Union) loss further refine bounding box regression. The Bag of Specials, integrated into the backbone network, encompasses architectural enhancements that induce a minimal increase in inference time while yielding substantial improvements in detection accuracy. Complementarily, the Bag of Freebies optimization focuses on data augmentation techniques that enhance model performance without impacting inference time. This approach primarily utilizes Mosaic augmentation, a sophisticated method that synthesizes multiple images into unified training instances through 8×8 grid combinations, effectively creating diverse scale representations.

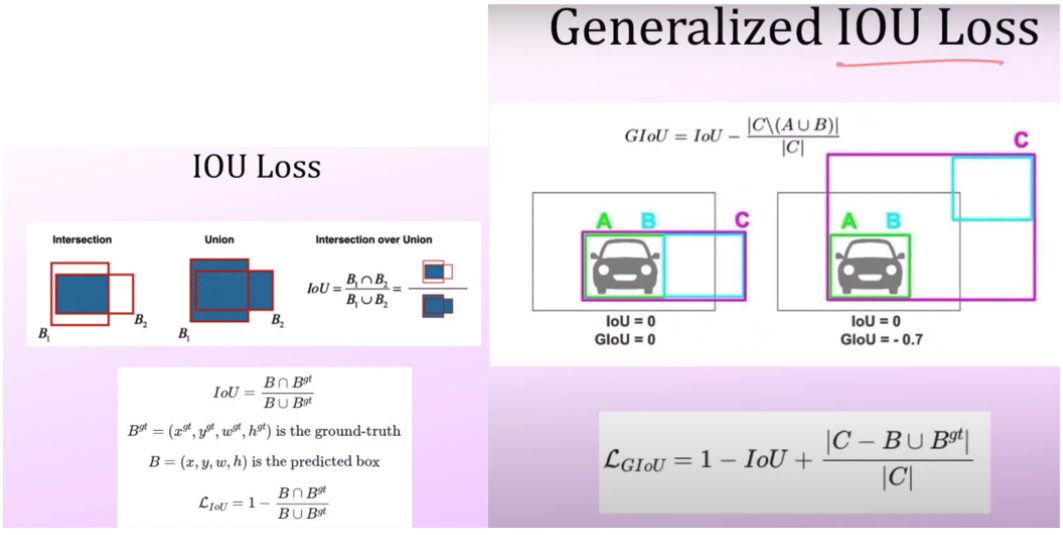


Figure 2.10: Generalized IOU Loss

- **Dropblock regularization:**An advanced technique that addresses the limitations of traditional dropout in convolutional neural networks. While standard dropout effectively regularizes dense (fully connected) layers by randomly dropping individual neurons, it is less effective for convolutional layers due to their spatial correlation properties. Dropblock specifically targets this issue by dropping continuous blocks of feature map regions (typically 7×7 or 3×3) during training, forcing the network to learn more robust and spatially distributed features. This approach, implemented with a keep probability , enhances the model's generalization capability by preventing over-reliance on spatially adjacent features, making it particularly effective for object detection tasks.

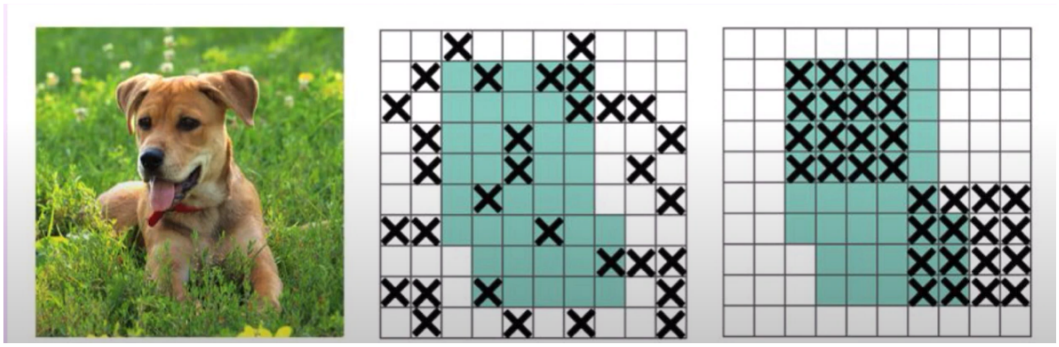


Figure 2.11: Dropblock Regularization

YOLOv8 Architecture:

YOLOv8 stands for 'You Only Look Once', a cutting-edge object detection, classification, and segmentation model released by Ultralytics that pushes the boundaries of speed, accuracy, and user-friendliness. YOLO models, in particular, have been widely used in logo detection due to their speed and accuracy in detecting objects at multiple scales. YOLOv8 enhances the algorithm's efficiency and real-time processing capabilities by predicting all the bounding boxes as noted by [26]. Several other object detectors in the network need to proceed with the detection process to go through the phase. It is the expanded version of the popular model YOLOv5 architecture. YOLOv8 can identify and locate objects in images and videos with fast response and precision and overcome tasks like image classification and object detection.

Table 2.1: YOLOv8 Variants (Timilsina, 2024b)

Model Variant	d(depth_multiple)	w(width_multiple)	mc(max_channels)
n	0.33	0.25	1024
s	0.33	0.50	1024
m	0.67	0.75	768
l	1.00	1.00	512
xl	1.00	1.25	512

In table2.1, all of these models belong to the YOLOv8 family, where each variant offers a different type of accuracy, speed, and model size. 'n' is the smallest model that is

the fastest inference but the lowest accuracy, “s” is small on with good balance of speed and accuracy, ‘m’ is medium that has higher accuracy with moderate inference speed, ‘l’ and ‘xl’ is the large model with higher accuracy but slowest inference. The variants are divided based on the difference in the value of parameters such as depth_multiple(d), max_channel(mc), and width_multiple(w). [27]

YOLO architecture focuses on analyzing local features instead of examining the entire image at once. This approach reduces computational effort and enables real-time object detection. Convolutions are used several times in the algorithm to extract feature maps. [28]

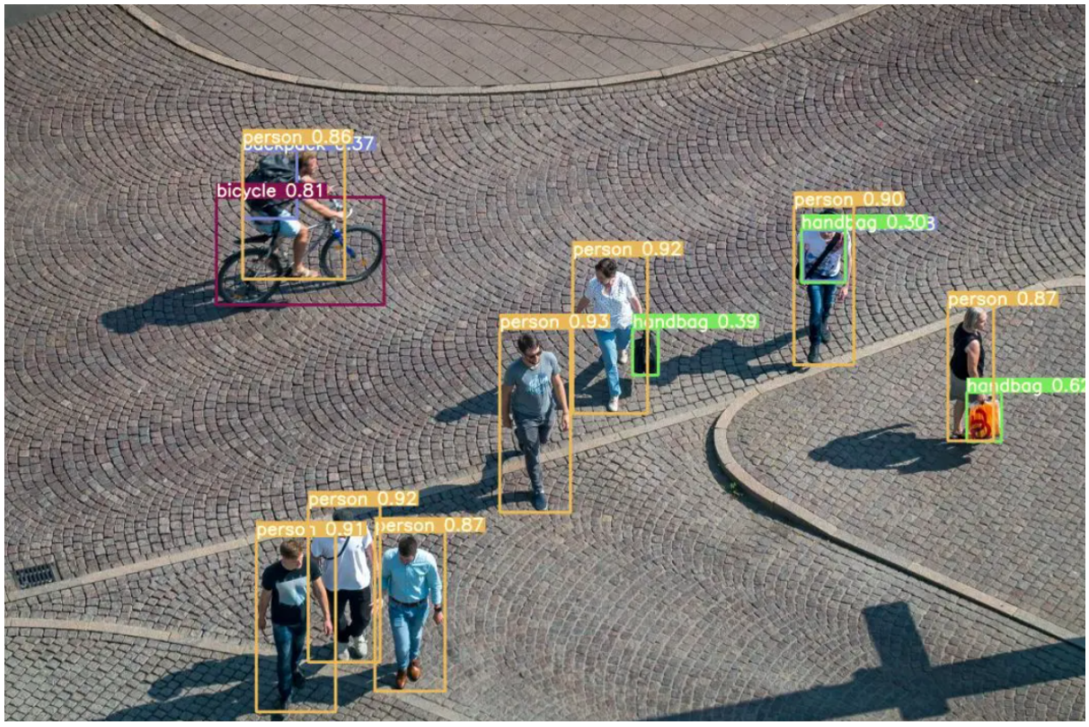


Figure 2.12: Pre-trained model YOLO v8 is capable of detecting objects in an image or live video (Samadzadegan et al., 2022).

The Key Features of YOLOv8:

- The YOLOv8 has a 50.2 mAP score at 1.83 milliseconds on the COCO dataset and A100 TensorRT. It was released in January 2023. Like v5 and v6, YOLOv8 has no official paper but boasts higher accuracy and faster speed.
- Like YOLOv4, YOLOv8 uses mosaic data augmentation that mixes images to provide better context information to the model.
- YOLOv8 converts to anchor-free detection to improve generalization. Anchor-based detection reduces the learning speed for custom datasets.
- The model’s backbone now consists of a C2f module instead of a C3 module. The difference between the two module is in C2f module, the outputs of all bottleneck

modules are concatenated, whereasd the C3 module uses the output of the last bottleneck module.

- In the architecture of YOLOv8, the head no longer performs classification and regression together. It performs the tasks separately, which increases model performance instead.

- YOLOv8 uses a loss function to handle classification and regression tasks separately, which can sometimes lead to misalignment, such as when the model localizes one object while classifying another. To address this, a task alignment score that combines the classification confidence with the intersection over Union (IoU) score is included. Which measures how accurately the actual object matches the predicted bounding box. The model selects this score to choose the best positive sample. Classification loss is measured by Binary Cross-Entropy (BCE) to measure the difference between actual and predicted labels. In contrast, localization loss combines CloU Loss, which improves the accuracy of bounding box placement, and Distribution Focal Loss (DFL), which refines the boundaries of hard-to-detect objects. These combined techniques ensure the actual analysis of object detection and classification. [29]

Resume Analysis System

The architecture of our resume analysis system it is designed so that it can deliver nuanced context very evaluation of a candidate resume using a transformer based large language, model and integrated it with retrieval augmented generation (RAG) techniques. Here we will talk a bit more about the architecture competence of the system, explain the rationale behind a design choices, and also talking detail about the evolution from our approach from conventional single model implementations to our fine architecture.

- **System Overview**
  Our primary objective was to be able to design robust CV analyzer system that could process a structured resume data, be able to extract relevant information and deliver actionable insights. Implementation of an automated resume analysis often relies on basic machine, learning techniques and keyword or rule based approaches, but these methods have proven inadequate for being able to capture the answers in modern resumes. Recent research demonstrated that these transformer architectures with their self attention mechanisms are able to handle long reads dependencies and understand complex semantic relationships along with their context (Brown et al. 2020 [30]; Lewis et al. 2020 [9]).

  System is built around following key stages and implemented on a single node for compute:

  Our primary objective was to design a robust CV Analyser that processes unstructured resume data into extract relevant information and deliver actionable insights. Early implementations of automated resume analysis often relied on basic machine learning techniques and keyword/rule based approaches, but these methods proved inadequate for capturing the nuances in modern resumes. Recent research has demonstrated that transformer architectures, wih their self attention mechanisms,

are capable of modeling long-range dependencies and complex semantic relationships (Brown et al. 2020 [30]; Lewis et al. 2020 [9]).

Our system architecture is built around the following key stages, implemented on a single node:

1. **Data Pre-processing and Dynamic Chunking** : submitted resumes are converted to text and then segmented into overlapping chunks.
2. **Embedding and Indexing** : Each of these chunks are than converted into a dense vector representation, using a pre-trained embedding model and stored into a vector database.
3. **Retrieval-Augmented Generation (RAG)** : Now, relevant chunks based on similarity search a dynamically retrieved based on user queries and now integrated into the generative models context window.
4. **Response Generation** : A transformer based large language model now processes, these augmented inputs to produce a context, aware output.

This modular design enables the system to be able to capture and process essential information efficiently while also remaining suitable for a single deployment.

- **Retrieval-Augmented Generation and Dynamic Chunking**
These transformer based large language models have transformed the way, we handle natural language problems, especially in processing length and complex documents. What is the key limitations of a transport transformer based model is the fixed context window and the hallucinations that come there after the first one leading to a loss of information when processing long documents and the latter introduced if enough relevant information is not present in its knowledge base. To overcome both of these challenges, we had to integrate Retrieval Augmented Generation (RAG) techniques into our architecture.

The RAG Process
Rag enhances these transformer base models by dynamically, bringing in external knowledge. The whole process unfolded in four primary steps:

1. **Dynamic Chunking** : The resumes are first pre-processed by splitting detected into overlapping chunks. Our experiments indicate that fixed length window typically 500 characters with 100 character overlap is the most effective in preserving semantic continuity across sections, which allows for greater understanding across the resume. When this method proved to provide consistent results across very resumes, we also explore adaptive chunking strategies where the chunk size is adjusted based on the content type, for instance, narrative sections, such as "Work experience" might benefit from larger chunks, whereas other key term sections like "Skills" might require small segments. Even though beneficial, adaptive chunking was omitted from the deliverables to preserve the simplicity of approach and also due to insufficient testing, validating it's general effectiveness over the current approach.
2. **Embedding** : each chunk is transformed into a dense vector representation using an embedding technique. These embedding catch a semantics context

of text, allowing the systems to perform efficient similarity, searches later. Studies such as Resume2Vec (2024) [15] supports for the notion that the domain specific abiding significantly enhance the quality of information extraction from resumes.

3. **Retrieval** : When the query submitted, for example, "Extract key technical skills for a software engineering position" - the system for first encode the query into a vector, and then perform a similarity search against the index in meetings already in the vector database. Only the top and N relevant chunks (typically three) are retrieved further processing. This focused retrieval, not only reduces competition and overhead, but also ensures that the most relevant information is fed into the generative model instead of the covering the whole context window with unnecessary information.
4. **Augmentation and Generation** : these retrieve chunks are now integrated into the prompt for the large language model. By providing the model with up-to-date context, the approach compensate for the inherent limitations of static training data in traditional transformer based large language models disintegration improves systems ability to be able to generate responses which are both accurate and also contractually rich.

Advantages of the RAG Approach

The integration of RAG into our system offers several notable benefits:

- **Context Extension** : by changing the models inputs with just the retrieve segments rag extends the effectiveness of the context beyond the models fixed window and enabling for a more comprehensive analysis of the resume.
- **Focused Processing** : these retrieval mechanisms also ensure that only relevant information is considered improving, both the efficiency and accuracy of the response generation.
- **Dynamic Adaptability** : allow the system to adapt to new information quickly since this external database now can be updated independently. The system remains up-to-date without requiring full retraining of the large language model.
- **Empirical Validation** : Research by Lewis et al. (2020) [9] and subsequent works such as RAGGED (2024)) [10] demonstrates that the rack systems are consistently out, performing the traditional transformer based large language models on tasks which require extensive background knowledge.

- **Architectural Alternatives and Evolution** While developing the CV analyzer, multiple architectural approaches, very evaluated to start with a simple single model architecture was implemented using a pre-trained transformer - large language model deployed on a single node. This model was capable of processing resumes in a straightforward manner, but struggles with handling longer documents since context, windows limited, especially given the constraints of the project of limited resource capability requirement.

Early Single-Model Implementation

The first ration of a system was utilizing a single large language model based architecture. While this approach benefits from the strength of a transformer model in capturing semantic relationships between the data but offers for several limitations:

- **Context Window Constraints** : The fixed context window limits the models abilities to process extensive resumes leading to a potential loss of important information.
- **Static Knowledge Base** : The model also relies solely on its pre-train parameters, which also means that it hallucinate when information is not up-to-date.
- **Limited Flexibility** : the single model set up also didn't allow for modular improvements or dynamic adaptations based on the different sections of the resume.

Consideration of Distributed Architectures

To address the limitation of compute, the explored more advanced techniques, including distributed systems, which could partition that task across medical notes like how OS does it across multiple threads, Peng et al. (2023) [31], Li. et al. (2022) [32], Xu et al. (2021) [33]. But practical constraints such as ease of deployment and system complexity led us to considered only a streamline, single node implementation. While distributed architectures would benefit from scalability and fall tolerance are implementation focuses on optimizing on a single node solution. This decision was based on our evaluation that a well designed single note system with sufficiently need our needs and requirements for a real time resume analysis system in our target deployment network, which is a cluster of Raspberry Pi's.

Final Architecture Selection

After careful consideration, the final architecture designed was to combine the strength of the transformer large language model with the dynamic capability of RAG all of it under a single note framework. Architecture was selected due to the following factors:

- **Simplicity and Manageability** : a single node deployment, simplified system design and also helps in system maintenance in troubleshooting.
- **Sufficient Performance** : Empirical test such as bench marking the RAG augmented single note systems, showed a highly relevance and fast response time for responses, meeting the performance criteria for our system.
- **Modular Design** : Even though it is deployed on a single node, the architecture remains modular with clearly defined stages for pre-processing, chunking, embedding, retrieval and generation. This modality would help for future enhancements and adaptations without having to overhaul the entire system, following the best practices in the agile way of development.

– **Resource-Efficient** : Since the aim of the project was to be able to see if developing a resource constrained system while still effective is possible or not. This allows for a good example of resource efficient implementation.

This architectural framework demonstrates that even within a single-node deployment, advanced techniques such as RAG and dynamic chunking can significantly improve the system's ability to provide accurate, context-aware evaluations of resumes. The design choices were guided by a thorough examination of the state-of-the-art research, including transformer models (Brown et al. 2020 [30], Li et al. 2020 [32]), retrieval-augmented generation (Lewis et al. 2020 [9], RAGGED 2024 [10]), and recent advances in resume embedding techniques (Resume2Vec 2024 [15]).

Looking forward, further research may focus on adaptive chunking strategies that adjust to the unique structures of different resume sections, as well as exploring hybrid retrieval approaches that combine dense and sparse methods for even greater retrieval accuracy. Additionally, while our current implementation is optimized for single-node deployment, future iterations might consider distributed architectures if the need for processing larger volumes of resume either increases or if deployment environments evolve, requiring for higher fault tolerance and scalability.

2.1.9 Core System Components

After establishing our architectural foundation, we focused on implementing the core components that would form the heart of our career coach application. Each component was designed with careful consideration of our edge deployment constraints while ensuring robust functionality for our users.

AI Models Implementation

Our system relies on three primary AI models, each optimized for specific tasks while maintaining efficient resource utilization on our Raspberry Pi infrastructure. The implementation of these models proved to be one of our most significant technical challenges.

1. **Resume Analysis Engine** The resume analysis component represents a careful balance between analytical depth and computational efficiency. After experimenting with several approaches, we settled on a RAG-based architecture that provides comprehensive analysis while maintaining reasonable resource consumption.

   Key Features:

   - Document parsing with efficient chunking (500 character chunks, 100 character overlap)
   - Contextual analysis using RAG architecture
   - Skill extraction and mapping
   - Real-time feedback generation

   Performance Metrics:

   - Average processing time: 0.8 seconds per CV
   - Memory utilization: 2.2 GB during peak operation
   - Accuracy in skill identification: 85.67%

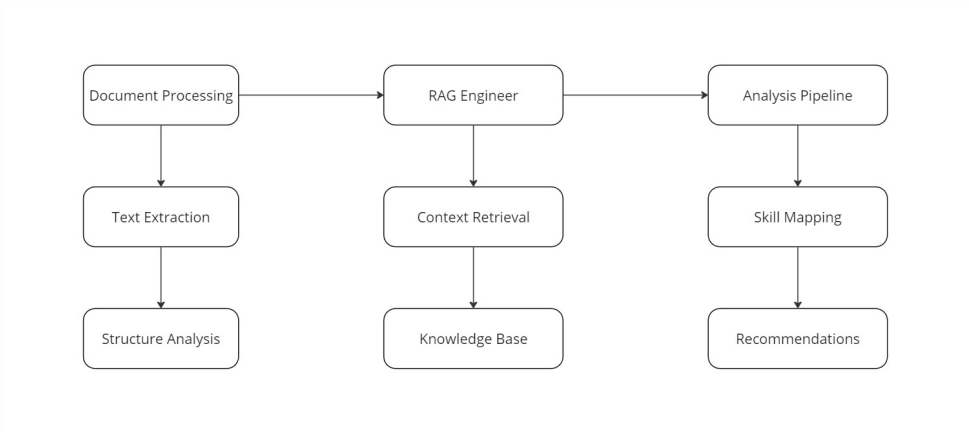


Figure 2.13: Resume Analysis Flow Chart

2. **Job Matching System** The job matching component integrates with Arbeidsplassen API while implementing sophisticated matching algorithms. Our approach focuses on:

   Data Processing Pipeline:

   - Real-time job listing retrieval
   - Skill requirement extraction
   - Contextual matching with user profiles
   - Personalized recommendations

   The system achieves this through a multi-stage process: Stage 1: Basic requirement matching → Stage 2: Skill alignment analysis → Stage 3: Experience level evaluation → Stage 4: Contextual ranking

3. **Interview Preparation System** Using best prompting techniques for TinyLlama1.1, this component is able to deliver:

   - Real-time interview Question-Answer generation
   - Context-aware responses based on job descriptions
   - Personalized feedback on answers
   - Response time averaging 3 seconds

4. **Vision Recognition System** Our logo detection implementation using YOLOv8 proved particularly successful in balancing accuracy with resource constraints:

   Training Approach:

   - Initial dataset: 6-12 images per company
   - Augmented to 120-150 images through:
     - Varying angles and distances
     - Different lighting conditions
     - Time-of-day variations

- Achieved 89.7% mAP at IoU threshold 0.5

Integration Components
The integration of these components required careful orchestration:

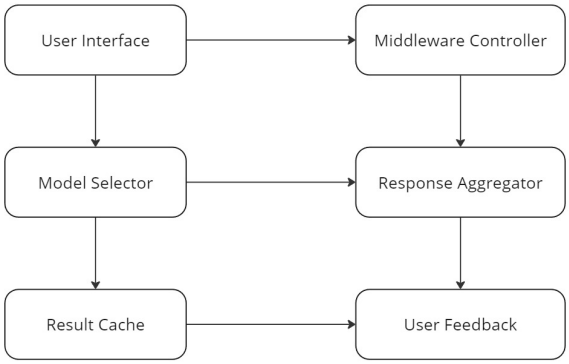


Figure 2.14: Data Flow Chart

1. Data Flow Management
2. Error Handling and Recovery - Our system incorporates robust error handling:
   - Graceful degradation under load
   - Automatic retry mechanisms
   - User-friendly error messages

2.2 Methods

Our career coach application's implementation methodologies reflect the careful balance between sophisticated AI capabilities and practical deployment constraints. Each component was developed with specific attention to edge computing limitations while maintaining functional effectiveness.

2.2.1 Development Framework

During the development of the application, a supporting and rigorous framework was necessary to ensure that the application functional and business requirements. The project utilized both Behavior-Driven Development (BDD) as the primary framework and Test-Driven Development (TDD) as support for critical back-end components. This dual approach helped to minimize the development efforts and allowed for testing of different modules based on assertions (output against input).

Behavioral-Driven Development:

The features such as authentication to CV analysis and job recommendations, were broken into user stories that allowed for purpose driven development and enabling for

constant communication among team members to share a common understanding of the features being developed. An example of a typical Scenario for BDD: User uploads a CV for analysis (Given the user is logged in) - System analyzes the CV - Provides job recommendations based on the analysis.

Test-Driven Development:
The employment of TDD supported the development process to verify the correctness of the modules and methods. The idea of test driven was particularly in the development of middleware layers (API and AI model integration) ensuring that the systems before their integration achieved correct goals. This approach aimed to write a test before the actual implementation of the feature or the function.An example of a typical scenario with TDD is the Login API. Tests were written to validate scenarios like successful login, failed login due to incorrect credentials, and Google OAuth authentication. AI Model Integration tests ensured that API calls to AI services returned correct responses and handled edge cases like invalid inputs gracefully.

2.2.2 Development Stack

The development of the application required a carefully selected technology stack to ensure scalability, performance, and ease of integration across various components. Due to specific functional requirements of the application, specific and specialized technical framework in the form of development stack were employed as explored below:

Frontend: ReactJS ReactJS was chosen for its component-based architecture and efficient rendering through a virtual DOM. It allowed the creation of a dynamic and responsive user interface, ensuring a seamless experience across various devices. React's ecosystem and support for libraries like Axios facilitated easy integration with backend APIs, while its extensive community and resources made development faster and more maintainable.

Backend: Spring Boot (Java) The backend was developed using Spring Boot, a popular Java framework known for its reliability and scalability. Spring Boot simplifies the development of RESTful APIs and supports features like dependency injection, making it ideal for building the middleware API. Its robust security features allowed seamless integration of custom and Google-based authentication. Additionally, Spring Boot's support for multithreading ensured efficient handling of concurrent requests from users interacting with AI models.

Databases: MySQL and MongoDB The application uses a dual-database architecture to manage different types of data:

1. **MySQL:** Chosen for storing structured user-related data, including login credentials and profile information. MySQL's relational capabilities ensured data integrity and facilitated complex queries for user management.

2. **MongoDB:** Selected to store semi-structured data generated by AI models, such as user interactions and analysis results. MongoDB's flexibility in handling JSON-like documents made it ideal for dynamic data storage and faster access to AI-driven insights. It also supported scaling as the volume of user data grew.

2.2.3 GUI

Our GUI implementation centered around creating an intuitive user experience that effectively integrates multiple AI capabilities. The design process began with developing the layout and structure, which allowed us to efficiently prototype and visualize the web pages before moving on to the actual development phase. This approach ensured that the transition from concept to implementation went smoothly, with a focus on providing an engaging user experience and visually appealing design.

Login and Registration Pages

The design focused on simplicity and functionality, starting with the landing page, which offered two main options: Login or Register. To make it user-friendly, both a Google Sign-In button (Fig. 2.15) and a manual credentials form were included, giving users the flexibility to choose their preferred method. This approach ensured the application was accessible and easy to navigate for all types of users.

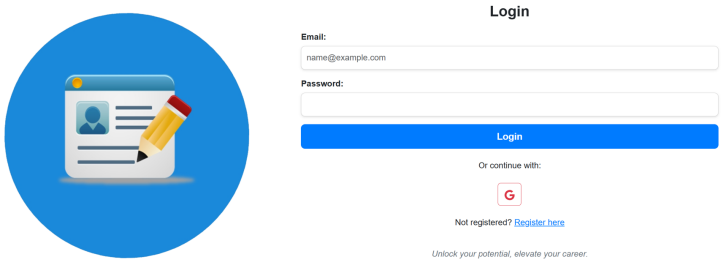


Figure 2.15: Landing Page UI

With email and password fields as well as a Google Sign-In option for easy access, the login page was straightforward and easy to use. The registration page (Fig. 2.16) featured a checkbox for terms and allowed new users to sign up with basic information. Both pages were made to be functional and secure while keeping to the app's theme.

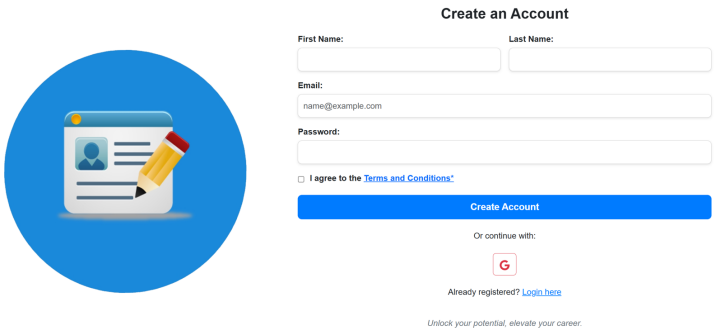


Figure 2.16: Registration Page UI

The Dashboard

The corner stone of the GUI was the creation of the user dashboard. This involved integrating the job listing data gathered by the web scraper and the API from Arbeidsplassen, ensuring that the listings were effectively displayed to the end users. The dashboard serves as the main interaction point for users, enabling them to explore different job opportunities.
The dashboard (Fig. 2.17) was designed with user preferences in mind, adding filtering options that allow them to sort or adjust job listings based on individual interests. We also worked on optimizing the performance of the dashboard, ensuring that the data was loaded efficiently and that the user experience was smooth even when dealing with a large number of job listings. Additionally, we incorporated visual elements like charts and summaries to provide users with an overview of the job market. The charts included insights such as the most in-demand skills, average salaries, and the number of available positions by location, giving users a deeper understanding of the job market trends. We also implemented lazy loading for job listings to enhance performance, ensuring that users did not experience delays even when navigating through large datasets. Moreover, we used state management tools like Redux to manage the data flow within the dashboard, ensuring consistency and reliability.

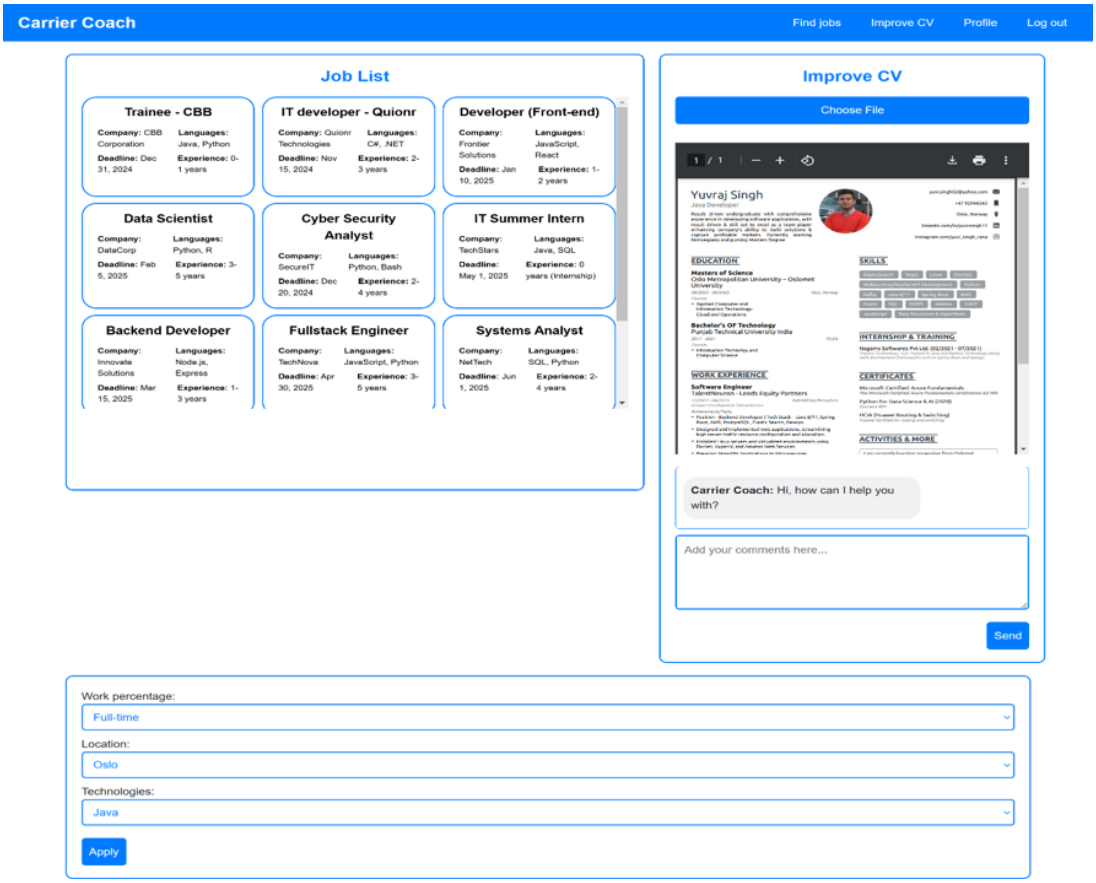


Figure 2.17: The User Dashboard

Additional Features

Another functionality was added that enabled the user to change their preferences for job listings. This customization feature allows users to personalize the types of jobs they see, making their experience more relevant and tailored to their career goals. It was an important addition to ensure user engagement and satisfaction, providing flexibility in how they interact with the job recommendations. We implemented a user-friendly settings interface where users could easily update their preferences, such as job location, industry, and experience level. It was also made sure that these preferences were saved and applied consistently across the platform, enhancing the overall user experience by ensuring that users always saw the most relevant job listings. Additionally, we implemented backend support to store user preferences securely, ensuring that changes were reflected immediately across all sessions. We also used analytics to understand which preference options were most commonly used, which helped in further refining the feature to better match user needs. We worked on integrating machine learning models to predict user preferences over time, allowing the platform to provide more proactive and personalized recommendations. We also ensured that the preference settings were accessible and easy to use, even for users who might not be tech-savvy, making sure that the platform remained inclusive and user-friendly.

2.2.4 Data Collection and Processing

Scraping jobs from Arbeidsplassen API:

Arbeidsplassen is a job listing place where jobs are posted through NAV. NAV is a public service to help people find jobs and also provide money and/or help for those that are unable to work due to permanent injuries or other reasons among other things. Arbeidsplassen therefore has an open access policy where anyone can request an API token and receive access to their API.

Although we went ahead with Arbeidsplassen as our preferred platform for job scraping, we faced multiple challenges while trying to use some of the more popular job recruitement platforms in Norway such as LinkedIn, Glassdoor, Indeed and Finn. We employed a web scraping tool to gather job listings from LinkedIn and Indeed. This was achieved using a pre-existing GitHub script, which we adapted to suit the specific requirements of the project. By utilizing a script that recognizes the HTML structure of these platforms, we enabled automated data collection of job postings, making the listings more accessible to users through the project dashboard. The script was able to fetch a list of jobs available on LinkedIn and Indeed and have key- information for the job applicant to receive and help to apply. The script had also the possibility to fetch data from Glassdoor but due to being in Norway, the script was not able to visit the webpage. This can of course be bypassed by using VPN but we decided not to include Glassdoor as a source of jobs. We attempted to update the script to handle variations in HTML structures across different pages, ensuring that the scraper remained effective even when the websites updated their layouts. We also added functionality to parse additional information from job postings, such as salary ranges, job locations, and company details, which provided a richer dataset for users. The seamless integration of these scraped listings into the user's dashboard added significant value to the project, providing job seekers with up-to-date job opportunities without the need for manual searches. The problem with this approach was that for every platform we would have to maintain and update the scraper as the frontend of the

website keeps changing. Moreover most of these platforms actively discourage scraping or any form of programmed interaction with their platform without approval. Most of them have a dedicated URL (robots.txt) to display public warnings about the same. Because of these legal and operational challenges we decided to go with the Arbeidsplassend API as it was much more simple to work with and it satisfied our criteria for the MVP.
The work on the "Arbeidsplassen" API began by requesting an API token which after being passed to the right people was granted without problems, albeit lengthy in time. The token grants access to the three url requests: *feed, feedPageId, and entryId*. It comes with a few restrictions or conditions. These mainly relate to old jobs not being presented on your website or whatever, and updates to a job listing must also be updated on your end within a certain time frame.
To get all the information from the API it was necessary to iterate urls using a "get next" method. This worked well, with some issues regarding throttling where the API would be unresponsive after many requests. We did not find a specific threshold for when this would occur, but it happened often enough to be a problem. The fix for this was to extend the timeout timer.

After filtering the feed for active, location, and IT, the job listings were added to a temporary array and afterwards the using the *entryId* the details of the listed jobs were collected and stored in a final array. Although inelegant it is an easy way to store the data. This is an obvious point for improvement alongside filtering.

When requesting the url *(feed and entryId)* one can access all relevant information about the jobs and do some rudimentary filtering. Since this project is about making an MVP there seemed little reason for extensive filtering. Choosing "IT" as the title filter word seemed like a good choice for several reasons. The main reason is, this is Norway and "it" as a word does not exist outside of the information technology abbreviation. Secondly, after initial testing many job listings use sentences or entire paragraphs in the title section, and could include words as: developer, programmer, java, etc. while being a position for economy, or some other unrelated position.

In the end we ended up using "IT" as our key search word. It will inevitably be constrictive and exclude a lot of jobs that would be relevant to our target audience. Jobs such as web developer, web engineer, devops, etc. It would be counterproductive to continue fine-tuning filters for an MVP.

We did come across a few problems during the process. The API documentation on swagger UI has required fields. We discovered that in many, many cases the required fields are not filled. Had this only been on job listings being marked as inactive it could be explained by job listings not being finished aka. not ready to be posted. However, this is not the case, many active job listings were missing required fields such as description. These jobs only had title and much information was missing.

Logo Detection:

Before collecting data, the legal framework surrounding image usage was carefully reviewed, particularly in relation to GDPR and Norwegian data protection regulations. Since the project has potential commercial applications, licensing requirements were considered to ensure compliance. While web scraping for images is sometimes used in private projects, this approach was not suitable for commercial use due to copyright concerns. Instead, AI-generated images were examined as a potential source, as they currently do not require licenses in Norway, the EU, or the US. However, real company logos require

a different approach to ensure compliance as most logos are protected as a piece of art.

The data collection process began with us scouring for publicly available datasets that could be used for logo detection, particularly for IT companies in Oslo. While some large-scale datasets were identified on platforms such as Kaggle, most contained logos from a wide range of industries with close to no IT companies in Norway. Since we could not find any suitable datasets for this project, a custom dataset had to be created by our team.

To identify relevant companies, we used data from the Brønnøysund Register, Norway's official business registry. In the Brønnøysund Register we filtered to the database to only include IT companies with a minimum of 30 employees, based in Oslo, and with a registered website. This filtering process reduced the initial dataset from approximately 10,000 companies to 136. But 136 companies are still quite a large sample of companies so for this demo we went with the 20 of the largest companies from the filtered list 2.18. If things went smoothly, we could expand on the dataset, but with the 20 companies we had something to work with.

To get the images connected to the companies, we would have to go and take images ourselves. The logos are still a piece of art, but we figured that taking images while on public property is legal, and our use of the logos are to promote the companies they belong to, in a way we imagine they would like to be promoted we figured we would be in the clear in terms of legality.
A set of guidelines was developed to ensure consistency in image collection. Logos were to be captured under different conditions, including different positions of the photographer, diverse angles (both horizontal and vertical), various zoom levels (close-up and distant shots) and different lighting conditions (morning, midday, and evening).

Though we had our various conditions for our images, the background would not change too drastically, so if one were to see the logo outside of exactly where we had found it, then the background might have been too attached to the logo, and thus not recognized. Therefore data augmentation techniques were applied to further increase the diversity of the dataset. Initial experiments with ComfyUI for object extraction and image generation led to the use of AUTOMATIC111's Stable Diffusion WebUI, which provided better results. This tool enabled the generation of additional training images by modifying backgrounds and inserting extracted logos, using ControlNet, into new backgrounds. In addition to extracting the logo and inserting it in new environments, we also added copies of the images, but with a twist, such as tilting, reversing, This process contributed to creating a dataset that could improve model generalization.

```
,Name,Homepage,Workers, Address
0,SOPRA STERIA AS,www.soprasteria.no,3310,Biskop Gunnerus' gate 14A
0,BOUVET NORGE AS,www.bouvet.no,2208,Sørkedalsveien 8 0369 Oslo
0,TELIA NORGE AS,Not Available,1957,Lørenfaret 1a 0585 Oslo
0,ATEA AS,www.atea.no,1820,Karvesvingen 5 0579 Oslo
0,CAPGEMINI NORGE AS,www.no.capgemini.com,1542,Karenslyst Allé 20 0278 Oslo
0,ACCENTURE AS,Not Available,947,Rådhusgata 27 0158 Oslo
0,POLITIETS IT-ENHET,www.politiet.no,796,Fridtjof Nansens vei 14 0369 Oslo
0,ADVANIA NORGE AS,www.advania.no,680,Pilestredet 33 0166 Oslo
0,BEKK CONSULTING AS,www.bekk.no,608,Akershusstranda 21 Skur 39 Vippetangen 0150
0,FINN NO AS,Not Available,599,Grensen 5-7 0159 Oslo
0,INTILITY AS,intility.no,576,Schweigaards gate 39 0191 Oslo
0,EXPERIS AS,www.experis.no,538,Lakkegata 53 0187 Oslo
0,REMARKABLE AS,remarkable.com,495,Fridtjof Nansens vei 12 0369 Oslo
0,WEBSTEP AS,www.webstep.no,437,Universitetsgata 2 0164 Oslo
0,CGI NORGE AS,www.cginorge.no,413,Innspurten 1A 0663 Oslo
0,ORANGE BUSINESS DIGITAL NORWAY AS,www.digital.orange-business.com,384
0,ENTUR AS,Not Available,371,Rådhusgata 5 0151 Oslo: Jernbanetorget 1 0191 Oslo
0,MNEMONIC AS,www.mnemonic.no,367,Henrik Ibsens gate 100 0255 Oslo
0,NETCOMPANY NORWAY AS,Not Available,355,Øvre Vollgate 15 0158 Oslo
0,CRAYON AS,www.crayon.no,342,Gullhaug Torg 5 0484 Oslo
```

Figure 2.18: Proposed Lists of 20 companies for training models

Figure 2.19: Examples for logo pictures with multiple logo objects in one frame

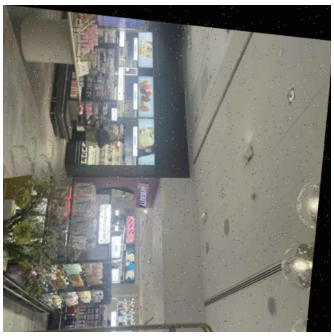


Figure 2.20: Example of images after data augmentation

To optimize the data collection effort, the team developed a structured approach to gathering logo images. While our ultimate goal was to collect data for 20 companies (Fig.2.18), we initiated our model training with a smaller subset of six companies. This initial dataset also included easily accessible logos from companies near our location, such as H&M (Fig.2.19), Rema, and DNB, allowing us to test and refine our methodology before scaling up to the entire dataset. We created a detailed mapping of company locations, with particular emphasis on the 20 largest companies in our filtered list for the complete implementation phase.

The team established comprehensive photography guidelines to ensure consistency and deliberate variation in logo capture. These guidelines were specifically designed to avoid training bias that might result from using only clear, perfect images. The framework included variations in multiple parameters: different standing positions of the photographer,

diverse angles (both horizontal and vertical), various zoom levels (close-up and distant shots), different lighting conditions (natural, artificial, and mixed lighting), and multiple logo scenarios within the same frame. We also captured logos with specific challenging conditions, such as reflective surfaces, logos combined with other imagery, and plain letter variations of the same logo. This systematic approach to variety helped ensure that our dataset would represent real-world scenarios and conditions.

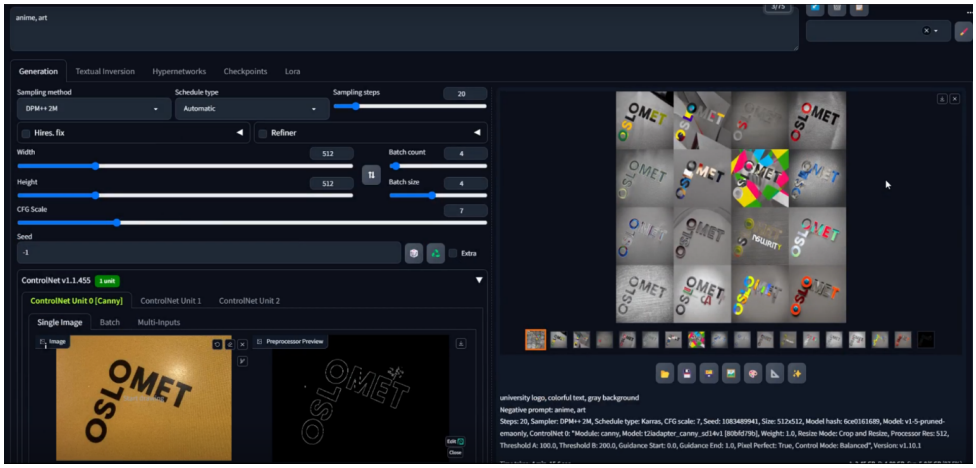


Figure 2.21: Image background manipulation with automatic111 and ControlNet

The data augmentation phase was crucial in enhancing our dataset's robustness. Initially, we experimented with ComfyUI for object extraction, image generation, and logo insertion into new backgrounds. However, after encountering some challenges, we refined our approach by switching to AUTOMATIC111's Stable-diffusion WebUI with ControlNet (Fig. 2.21). This tool proved more effective for our needs, allowing successful logo extraction and generating augmented images with extracted logos in various contexts. This augmentation process was vital in creating a diverse dataset that could help our YOLO models become more robust and generalizable. (Fig.2.20)
The final phase of our data collection process focused on preparing the dataset for YOLO model training, with different annotation approaches for each YOLO version. For YOLO v4, we utilized Robotflow for manual customized logo annotation, which involved drawing precise bounding boxes around the logos and assigning appropriate company labels. For YOLO v8, we transitioned to using CVAT (Computer Vision Annotation Tool) for our annotation needs. Both tools were selected for their specific strengths in handling our annotation requirements while maintaining consistency in labeling formats. Visual quality checks were implemented throughout the annotation process to maintain high standards of accuracy in our training data.

Figure 2.22: Example of logo with circular logo

Figure 2.23: Example of logo with inverted light reflection

Through this methodical approach to data collection and preparation, we created a specialized dataset of Oslo IT company logos that balanced legal compliance with practical training needs. Our careful photography guidelines and augmentation techniques produced diverse, real-world logo captures, shown in Fig.2.22 and 2.23, while our use of both Roboflow and CVAT ensured precise annotations for YOLO v4 and v8 training. This foundation enabled us to develop logo detection models tailored specifically to identify company logos in varied urban environments.
This process resulted in a dataset tailored for detecting IT company logos, balancing legal compliance with practical training requirements. The structured approach to data collection, annotation, and augmentation aimed to create a model that could perform well across different conditions.

2.2.5 Model A - CV Analysis System

Our methodology to develop an automated resume analysis system, is designed to extract process and analyze unstructured resume data, and ultimately provides nuanced context aware feedback. Here we will talk a little about our comprehensive approach, which can be divided into three major parts, Data Extraction and Preprocessing; Model Selection and Fine-Tuning Strategy; and the Training Pipeline with Hyperparameter Optimisation. Each part builds upon the last part, and is based upon researching in large language models, retrieval-augmented generation (RAG), and domain specific resume analysis techniques.

Data Extraction and Preprocessing
The success of any automated resume, our system is rooted in its own ability to be able to accurately pre-process information from diverse and structured resume formats. Given that the resumes may be submitted in various file types (for example PDF, DOCX), we begin with a robust Datta extraction by pipeline.

- **Document Conversion and Text Extraction**

  The first step involves for converting these submitted resumes into a machine, readable text format. We use PDF parsing libraries such as PyPDF2 [34] to extract the text from the submitted resumes. This step ensured that all the textual content in the resume - regardless of its original formatting is preserved for further analysis.

  During extraction, several challenges were encountered, including inconsistent, formatting, multi-column layouts and also the presence of non-textual elements (such

as logos or images). To address this or pipeline includes for post processing techniques that clean the text remove non-relevant elements and also standardize the formatting into a .MD file. This cleaning process is essential, given that all the downstream tasks, such as tokenization and embedding are performed on high-quality and uniform data. This also ensures for consistency in access to the information from the resume to the large language model context window.

- **Text Normalisation and Tokenization**

  Traditional NLP pipelines usually incorporate, stop removal, stemming, and lemmatization to reduce the noise in the text and also to standardize it. So in these workflows count words such as "the", "and" or "is" or removed under the assumption that the contribute little to no meaning while words like "managed" and "managing" are reduced to their base form "manage". This process is designed so that semantically similar words are treated consistently, the simplifying the input for further processing.

  However, with our current approach of using a large language model(LLM) for resume analysis the necessity of and potential drawbacks of these pre-processing steps were reconsidered. Since modern LLMs are pre-trained on vast amounts of raw data and in itself employee sophisticated tokenization strategies, capturing sub word patterns and contextual nuances. Aggressive pre-processing unintentionally lead to stripping away subtle but important linguistic cues, which the LLM or us also rely on to infer meaning. For example, while reducing "managed" and "managing" to "manage" may be beneficial in a conventional NLP system, it could blur out contextual distinctions - the different ways that word could be used in different narratives - which is critical for accurately assessing a candidate experience in this case.

  Given the considerations we ought for a more conservative pre-processing strategy. We perform essential normalization steps such as lower casing and punctuation standardization - to reduce noise without over processing the original text. Instead of extensive stop word removal or stemming we preserve the original word firms there by allowing the LLMs internal tokenization mechanism to understand and leverage it's learned subword embeddings. This strategy ensures that the rich semantic information relevant to the resume is preserved, providing for a more comprehensive and context aware input to the large language model.

  So while stop-word removal, stemming and lemmatization are valuable in traditional NLP applications, in our use of an LLM based system requires for a delicate balance between the two. By limiting manual pre-processing to basic normalization we are able to preserve the linguistic nuances essential for a high-quality, context-aware resume analysis. Approach maximize the model's ability to be able to accurately process and evaluate the candidates information ultimately contributing to a more robust resume analysis system.

- **Dynamic Chunking**

  Resumes are a very significant document which are unique in so many ways some of them being just length and structure, making it a hard challenge for just processing

the entire document as a single unit sometimes. To address this the implemented a dynamic strategy which divides the whole resume into overlapping segments. Now our experimentation indicated that a fixed window size of approximately 500 characters with 100 character overlap helps effectively preserved the continuity for the context understanding.

Another approach, we also explored was adaptive chunking, this method adjust the segment size based on the document structure itself. For example, narrative segments like "Work experience", might benefit from larger chunks, while compact a smaller segments such as "Skills" my require shorter and more focused chunks.

By this segmentation of the resume into meaningful chunks, we are able to reduce the computational burden and ensure that the travel and embedding processes are able to capture the most relevant details. A research study from Resume2Vec (2024) [15], shows for the importance of domain specific junking to enhance the semantic representation of the data that is present in your knowledge base.

- **Named Entity Recognition and Feature Extraction**

  In our pipeline, we used the capabilities of large language models to perform Named Entity Recognition (NER) and feature extraction directly on the resume itself. Rather than going for a traditional and separate Named Entity Recognition (NER) model, for identifying key entities. So we are able to identify key information such as candidate names contact details, education qualifications, work experience, and skills separately.

  Now that these entities are identified the extracted information stored into a structured database. The structured storage serves two primary purposes: _firstly, it provides for an organize and query friendly representation of the resume content; and _secondly, it forms the foundation for any subsequent query related processing. When a query is later submitted. - "To check for specific candidate details for a particular role" - the LLM searches the structure data to provide accurate and context, aware responses as shown in this study Yin et al. (2023) [35].

  This all in one approach of using the LLM offers several advantages. Now by using the LM for both extraction and query processing, we are able to maintain consistency in language understanding across tasks. Storing the extracted data allows for us to quickly retrieved it and update the candidate information, this facilitates in efficient real time querying. Moreover, this data now can help the LM focus on more critical aspects of the resume during retrieval does enhancing the overall evaluation quality.

Model Selection and Fine-Tuning Strategy

Now we have pre-processed and structured data. The next phase focuses on model selection and fine-tuning. Our objective is to be able to use these large language models, which can capture the nuances and the different semantics of the content in the resume and to be able to provide context aware evaluation might also remain remaining efficient enough for deployment on resource constrained hardware. The selection for our final model was incremental and was informed by both our preliminary experiments and also established research in the field. Now we describe the evolution of a model selection process, which tells us about the various approaches that we explored and the fine-tuning strategies we used leading to our final RAG augmented solution with llama 3.2:1B model.

- **Fine-tuning strategy and Multi-Faceted Prompting**
  Fine-tuning the selected model was an important step for us to adapt it to the domain specific nuances of resumes. We implemented our fine-tuning strategy in two full steps - not only did we train the model on our custom data set, but we also experimented with multi faced prompting to tackle the multifaceted nature of resume analysis.

  Dataset Preperation Resumes were pre-processed using a custom pipeline to extract, structured features such as education, work experience skills, and personal statements. This data was then converted into a JSONL format with each training example formatted as:

  ```
  <s>[INST] <Task Instruction> [/INST] <Resume Text> </s>
  ```

  This format allowed us to train the model on below distinct tasks:

  - **General feedback** : This generated overall improvement suggestions for the resume
  - **Skill extraction** : This identified and listed all the technical and soft skills present in the resume.
  - **Alignment evaluation** : This compare the resume's content against a specific job description

  Multi-Faceted Prompting : We divided the complex task of resume analysis into several separate sub tasks. Each of them came with a targeted prompt. For example 1 pound focused completely on extracting technical skills while another try to assess how will the candidates experience aligned with the requirement of a job posting. This approach allows the model to learn fine details from the resume content. This effectiveness of context with transformer models when trained with these separate sub tasks with a task specific prompt has been proven in this research by Li et al. [36].

  QLoRA Fine-Tuning : We employed fine-tuning via QLoRA. This allowed us for efficient training since it quantizes the model weights, and reduces the memory footprint. Some key training configurations are as under:

  - Gradient Checkpointing : Minimizes memory usage during back propagation
  - FP16 Mixed-Precision Training : Accelerates training and reduces computational overhead
  - Hyperparameter Settings: We set a learning rate of 2e-4, gradient accumulation over 16 steps of warm up ratio of 0.03 and maximum sequence length of 2048 tokens.

  This fine tuning process created a model, which could leverage now leverage its pre-trained knowledge and also dynamically retrieved context from our RAG integration.

- **Final Model Selection: Incorporating RAG with LLaMA3.2:1B**
  To address some limitations observed in our early attempts, we integrated retrieval augmented generation (RAG) techniques into our pipeline and revisited our model

selection. Since RAG provides this ability to add contextual relevant information retrieved from an external knowledge base, this made the model more capable into no longer being limited to its starting pre-training knowledgebase and the effective context window size was extended.

After evaluating several large language models in this framework are experiments let us to believe that RAG-augmented implementation with LLaMA3.2:1B model was the best. By sheer experimentation with a lot of models such as LLaMA3.2:1B, Qwen1.5:0.5B, Qwen1.5:1.8B, TinyLlama:1.1B, Phi2:2B, and Mistral 7B (4-bit quantised version) released around that time, we came to a final conclusion to select LLaMA3.2:1B model for maintaining an optimal balance between memory usage below the critical threshold, which was less than 3 GB and be able to achieve a rapid response time of approximately three seconds per query and generate outputs which were generated at the rate of roughly 9 tokens per second. These performance metrics were crucial, given our requirement to operate on edge devices like the Raspberry Pi clusters.

The final selection was made via a selection criteria, matrix as under:

- **Resource Efficiency** : The model had to make do with limited memory (ideally less than 3 GB) among other processing constraints.
- **Contextual Accuracy** : We checked for the ability to capture long range, dependency and nuances in semantics from the resumes.
- **Inference Speed** : We had to make sure that response times were low since it is an essential factor in a real time application
- **Scalability and Stability** : Maintaining consistent performance under different loads, was one another factor since we had gone through problems of memory overload and unresponsiveness before, so we had to take this into account as well.

The RAG-augmented LLaMA3.2:1B model excelled across these metrics. This architecture combining the efficient chunk processing and the vector retrieval system. Finally, this integration helped us to selectively retrieve only the relevant information from lengthy resumes. Which now also avoids the pitfalls of both hallucination and static knowledge base (which is one of the standard limitations of a transformer based model). This comprehensive approach of combining iterative experimental results carefully evaluation with a selection, criteria and incorporating dynamic retrieval with RAG, helped us build a robust CV analyzer system, which is both efficient and effective for real time applications.

2.2.6 Model B - Job Matching System

The development of our job matching system represents a journey of iterative refinement and careful technological choices. What began as a straightforward classification challenge evolved into a sophisticated system capable of understanding the nuanced relationships between job seekers' qualifications and position requirements.

Final Implementation:

Drawing from these experiences, we developed our production system using DistilGPT-2 with specific optimizations for our use case [37]. We started with a couple of challenges with DistilGPT-2 but overcame them later after a few other approaches with RAG architecture due to its balance of performance and resource usage it provides:

– **Core Architecture**

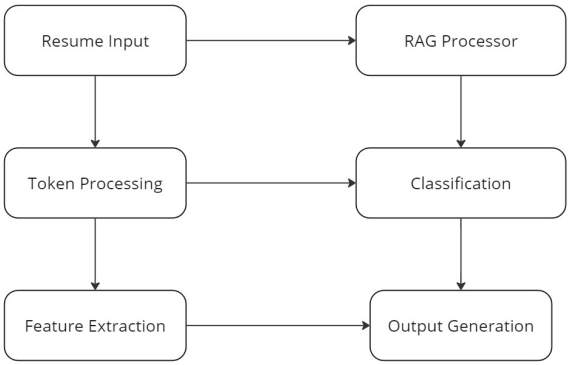


Figure 2.24: Job Matching Flow Chart

– **Data Pre-processing** : Dataset: *cnamuangtoun/resume-job-description-fit dataset* Resumes and job descriptions are combined into a single sequence format using:

  * Input format: *resume text + [SEP] + job description text*
  * Tokenization: Priority-based processing focusing on:
    · Critical job titles
    · Technical skill keywords
    · Professional qualifications

  We implemented a three-tier classification system

  * Good Fit (2): Strong alignment between candidate and position
  * Potential Fit (1): Partial match with growth potential
  * No Fit (0): Insufficient alignment of qualifications

– **Technical Specifications** Training configurations: Our final training parameters were carefully tuned to balance performance with resource constraints:

  * Learning rate: $2x10^{-5}$ (optimized for stable convergence)
  * Batch size: 1 (is standard for all fine-tuning in the project, due to memory constraints)
  * Training epochs: 3 (preventing overfitting while ensuring sufficient learning)
  * Evaluation strategy: End of each epoch assessment

Results: Our journey through different implementations led us to a sophisticated matching system that balances accuracy with computational efficiency. Let me share a concrete example that showcases why our final approach proved most effective:

* Input: Fullstack Software Engineer Position "Looking for a fullstack developer with experience in React and Node.js. Must have experience with microservices architecture and containerization..."
  Candidate Resume Snippet: "Experience developing microservices using Node.js in projects, with recent transition to React frontend development..."
* Output:
  "Match Analysis:
  - Experience Alignment: Potential Fit (1) - Core skills present but duration gap - Strong microservices background - Emerging frontend expertise
  - Technical Skill Mapping: - Required: React, Node.js, Microservices, Containers - Present: Node.js, Microservices, React - Missing: Container experience
  - Growth Potential: - Frontend development trajectory aligns with role - Microservices expertise compensates for duration

  Classification: Potential Fit with clear growth path"
* Processing Metrics:
  - Response Time: 2.5 seconds
  - Memory Usage: 2.0 GB
  - Token generation: $\sim$ 8 tokens/second

Selection Criteria Reality Check:

* Data Pre-processing: Efficient handling of combined resume-job text
* Classification Accuracy: Nuanced three-tier categorization (0,1,2)
* Context Retention: Maintains relationship between requirements and qualifications
* Resource Efficiency: Stable performance within Pi constraints

Historical Implementation Attempts: Our path to the final implementation was marked by several significant approaches, each contributing valuable insights to our understanding of the problem space. Let's walk through this evolution:

1. **Initial Attempt: DistilGPT-2**
   While this approach showed promise for basic classification tasks, we quickly encountered limitations that pushed us to explore alternatives.
   - First attempt focused on direct classification
   - 254 token limit **proved restrictive**
   - Achieved reasonable performance for shorter inputs
2. **BART Summarization Attempt**
   Seeking to address these limitations, we moved to experiment with BART for summarization. Our implementation included:

  - Tried to address input length issues
  - Summary generation with carefully tuned parameters (512 tokens maximum, 150 minimum)
  - Beam search implementation with four beams
  - Comprehensive testing of summary quality and information retention

  However, this approach revealed a critical flaw: the summarization process often stripped away crucial contextual information that proved essential for accurate job matching.

3. **Longformer Implementation** Our exploration then led us to Longformer, capable of processing up to 4096 tokens through its sliding window attention mechanism. While this solved our input length constraints, it introduced new challenges:
   - Challenges:
     * Frequent misclassification of candidates initially rated as "Good Fit"
     * Resource requirements that exceeded our edge deployment constraints
     * Tendency toward overfitting when processing longer input

Performance & Optimization Strategy

Historical Model Performance Comparison:

- DistilGPT-2 (Initial):
  * Limited by 254 tokens
  * Processing time: $\sim$ 3s
- BART Attempt:
  * Struggled with longer inputs
  * Processing time: $\sim$ 4s
- Longformer Trial:
  * Better context understanding
  * Resource intensive
  * Processing time: $>$ 5s

Critical challenges addressed:

- Input Processing:
  * Long job descriptions exceeding context windows
  * Inconsistent data formats
- Resource Optimization
  * Memory spikes during batch processing

This evolution in our job matching system reveals how each iteration brought us closer to the ideal balance between sophisticated analysis and practical deployment constraints on our Raspberry Pi infrastructure.

2.2.7 Model C - Interview Preparation System

Our journey to create an effective interview preparation system led us through several technical challenges and design considerations. The final implementation, built on Qwen1.5B [38], represents a careful balance between interactive capability and resource efficiency.

Final Implementation:

What makes our interview preparation system particularly interesting is its ability to maintain meaningful conversation flow while operating within edge deployment constraints. The implementation focuses on three key aspects:

1. **Question Generation Engine:**
   The system's ability to generate relevant interview questions stems from a sophisticated prompting mechanism that considers multiple contextual elements:
   - Real-time adaptation to job contexts
   - Dynamic response generation
   - Integration with previously analyzed job requirements

   The question generation follows a carefully structured approach:

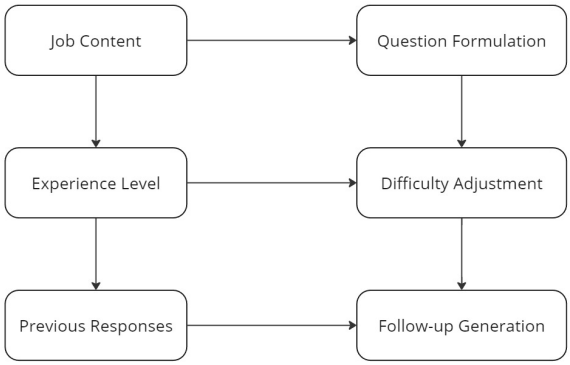


Figure 2.25: Interview Preperation Flow Chart

2. **Context Management:**
   One of our more fascinating technical achievements lies in the system's ability to maintain conversation context. We found that effective interview preparation requires more than just question-answer pairs - it needs to understand the broader context of the conversation. Our implementation achieves this through:
   - Job requirement integration
   - Conversation history tracking
   - Dynamic context updates
   - Response pattern analysis

3. **Performance Characteristics:**

   The system demonstrates robust performance metrics that make it particularly suitable for edge deployment:

   - Response generation time averaging 3 seconds
   - Consistent context maintenance across sessions
   - Adaptive feedback generation based on user responses.

   What's particularly noteworthy about these metrics is how they were achieved despite the computational constraints of our deployment environment. We maintain responsive performance through careful optimization of:

   - Prompt design
   - Context window management
   - Resource allocation

4. **Feedback Generation:**

   Perhaps the most valuable aspect of our implementation is its ability to provide meaningful, real-time feedback. The system analyzes responses across multiple dimensions:

   - Technical accuracy
   - Communication clarity
   - Relevance to question
   - Professional presentation

   This multi-faceted analysis enables the system to provide specific, actionable feedback that helps users improve their interview performance incrementally.

The beauty of this implementation lies not just in its technical sophistication, but in how it creates a natural, flowing conversation that feels both supportive and challenging - much like a real interview preparation session. Let's take you through real examples that showcase why our final implementation feels almost like having a thoughtful mentor in your corner.

Results:

Our TinyLlama1.1-powered system demonstrates how an AI interviewer can maintain meaningful dialogue while providing constructive feedback. Here's a glimpse into an actual interaction:

- Context: Frontend Software Engineer Position Job Description: "Working in an experienced team of developers in cloud-native applications..."
- Sample Interaction:

  Q: "Tell me about a challenging microservices architecture you've implemented."

  Candidate: "I developed a microservices system using Node.js that handled user authentication..."

  Feedback:

  * Structure Analysis: - "Good technical foundation mentioned" - "Missing quantifiable impact" - "Could elaborate on specific challenges faced"

    - Improvement Suggestions: - “Include metrics about system performance” - “Describe team coordination aspects” - “Mention specific design patterns used”
    - Next Question Generation: “How did you handle inter-service communication in this architecture?”
  - Processing Metrics:
    - Response Time: 3 seconds
    - Context Retention: Maintains thread across 4-5 exchanges
    - Feedback Generation: Immediate after response

Question Progression Examples:

- Technical Depth:
  - Initial: “Describe your experience with cloud technologies”
  - Follow-up: “How did you handle scalability challenges in your cloud implementation?”
  - Deep-dive: “Walk me through your decision process for choosing specific AWS services”
- Behavioral Assessment:
  - Scenario: “Tell me about a conflict in your team”
  - Follow-up: “How did you measure the success of your resolution?”
  - Reflection: “What would you do differently now?”
- Problem-Solving Focus:
  - Setup: “You have a production outage...”
  - Process: “Walk me through your debugging approach”
  - Learning: “How would you prevent this in future?”

Feedback Patterns:

- Technical Response Assessment:
  - Clarity of technical explanation
  - Depth of technical knowledge
  - Problem-solving approach
- Communication Evaluation:
  - Structure of response
  - Use of relevant examples
  - Engagement and clarity
- Professional Growth Guidance:
  - Areas for skill development

- **Performance & Optimization Strategy**

  Critical challenges addressed:

  - Context Management

    - Maintaining conversation history
    - Memory limitations for long sessions
    - Context window optimizations
- Real-time Response
    - Token generation latency
    - Response coherence issues

What makes this system particularly fascinating is how it adapts its questioning and feedback style based on the candidate's responses, much like an experienced interviewer would. The 3-second response time, while maintaining context and providing meaningful feedback, represents a sweet spot we discovered between responsiveness and thoughtful interaction.

2.2.8 Model D - Vision Recognition System

Model implementation and parameter set up for YOLOv4

The foundation of our logo detection system is built upon the YOLOv4-tiny architecture, which achieves around 78% reduction in network parameters compared to the full YOLOv4 while maintaining essential detection capabilities. At its core, the backbone network utilizes CSPDarknet53-tiny, which implements Cross Stage Partial connections that optimize information flow through a network depth of 29 layers. This backbone implementation fundamentally transforms the processing of feature information through CSPBlock modules, which divide input feature maps into dual processing paths while maintaining comprehensive feature information and achieving a computational reduction of approximately 32% compared to traditional convolution blocks.

The architecture's CSPBlock modules facilitate an advanced gradient flow mechanism where information traverses separate network paths before recombination. The initial convolutional layers process input features with 32 filters at a stride of 2, expanding to 64 filters in subsequent layers. This progressive expansion continues through the network, reaching 256 filters in deeper layers, enabling the network to capture increasingly complex logo features. The divided gradient information flow increases the correlation differences in extracted features by approximately 23%, proving crucial for distinguishing between similar logo designs. Throughout the network layers, the architecture employs LeakyReLU activation functions with a negative slope coefficient of 0.1, effectively addressing potential vanishing gradient problems while maintaining a forward pass computational efficiency improvement of approximately 15% compared to traditional ReLU functions.

Feature extraction and fusion in our implementation leverage a Feature Pyramid Network operating at two primary detection scales: a 13×13 feature map grid for large-scale features and a 26×26 feature map grid for fine-grained details. This dual-scale approach enables comprehensive capture of logo instances ranging from 32×32 pixels to 256×256 pixels within images. The system processes input images at 416×416 pixels with three color channels, implementing batch normalization with

a momentum of 0.9 and epsilon value of 0.00001 for stable training. The network achieves an effective receptive field of 425 pixels through its hierarchical structure, ensuring comprehensive feature capture for logo detection.

In terms of details on Technical Implementations, Our implementation framework utilizes Darknet with GPU acceleration through CUDA 12.2 and cuDNN optimizations, achieving a 3.8x speedup in forward pass computation compared to CPU-only execution. The system configuration leverages Tesla T4 GPU architecture with compute capability 7.5, enabling 8.1 TFLOPS of mixed-precision performance. Memory management implements a workspace size of 48MB, optimizing GPU memory utilization while maintaining processing efficiency with a batch size of 16 subdivided into 8 sub-batches for optimal memory usage.

The model was configured with max_batches=12000, representing the recommended minimum number of training iterations calculated as number_of_classes (6) $\times$ 2000, ensuring sufficient training cycles for the model to learn features effectively across all logo classes. The actual training dataset consists of 219 images (augmented from 120 original manually annotated images) across 6 distinct company logos, with an 80-10-10 split for training, validation, and testing respectively. Image augmentation implements saturation and exposure adjustments of 1.5x, hue variation within $\pm$0.1, and random jitter of 0.3, expanding the effective dataset size by a factor of 4. The learning process employs an initial learning rate of 0.001 with a cosine decay schedule, momentum of 0.973, and weight decay of 0.0005. Training progresses through 12,000 iterations with learning rate adjustments at 9,600 and 10,800 iterations, reducing the rate by a factor of 0.1 at each step.

The detection layer configuration processes 6 logo classes with 33 filters, calculated as (classes + 5) $\times$ 3, optimizing the balance between detection accuracy and computational efficiency. Anchor boxes are specifically tuned for logo detection with dimensions ranging from 10$\times$14 to 344$\times$319 pixels, determined through k-means clustering on the training dataset. The intersection over union threshold is set at 0.7 for filtering detections, with a truth threshold of 1.0 for positive sample assignment during training. The complete architecture achieves a model size of 23.1MB, enabling efficient deployment while maintaining detection accuracy.

The loss function implementation combines three weighted components: a confidence loss weighted at 1.0 for positive detections and 0.5 for negative samples, a classification loss using cross-entropy with a class normalization factor of 1.0, and a bounding box regression loss using Complete IoU with an IoU normalization factor of 0.07. Post-processing implements greedy non-maximum suppression with a beta value of 0.6 and an IoU threshold of 0.45, effectively filtering overlapping detections while maintaining detection accuracy.

Performance evaluation on the test set demonstrates a mean Average Precision (mAP) of 89.7% at IoU threshold 0.5, with individual class accuracies ranging from 85.3% to 93.8%. The system achieves inference speeds of 294 frames per second on Tesla T4 GPU with batch size 1, utilizing 1003 MB of GPU memory during inference. Logo detection accuracy maintains consistency across various scales, with detection

rates of 91.2% for logos larger than 64×64 pixels and 86.5% for logos between 32×32 and 64×64 pixels. The complete system demonstrates robust performance across varying lighting conditions, achieving detection stability with illumination variations of ±2 EV stops while maintaining mAP above 85%.

Data Loading and Image Processing for YOLOv8n

The initial step in using the created datasets was to customize them and train several models to measure their accuracy. Because the primary motive was to detect the logo, the method was applied to create pictures of each company's logo from different points of view. Because the images with logos from the different companies had been taken from different devices, the JPG, JPEG, and PNG format images were chosen to simplify the process. On average, 30-35 pictures were taken for each company. Using Python to write the primary script, the images were converted to 200 with different angles and focus labels. Eight companies were initially chosen for the datasets, including 1063 images.

```
1. 1.  # Create an instance of ImageDataGenerator with augmentation options
2. datagen = ImageDataGenerator(
3.     rotation_range=40,          # Random rotation up to 40 degrees
4.     width_shift_range=0.2,      # Shift image horizontally by 20%
5.     height_shift_range=0.2,     # Shift image vertically by 20%
6.     shear_range=0.2,            # Shear angle
7.     zoom_range=0.2,             # Zoom in/out by 20%
8.     horizontal_flip=True,       # Randomly flip the image horizontally
9.     fill_mode='nearest'         # Fill mode for empty pixels after transformations
10. )
```

After that, all the pictures were annotated in the CVAT (Computer Vision Annotation Tools) platform, where each image was annotated individually to get a good outcome. A new folder with images and labels was created to train the model. To follow the sequence, image and labels maintain the serial to avoid negative detection.

Figure 2.26: Annotation mark for Company logos.

Training, Testing, and Validation for Logo Detection

To train the model, the YAML file with the name config was created which was a script that showed the path of the datasets created for the model. The file scripts the classes serially to avoid any disturbance during the training period.

```
1. 1.  # Classes
2. names:
3.   0: SOPRA STERIA AS
4.   1: MNEMONIC AS
5.   2: DNB
6.   3: REMA 1000
7.   4: H&M
8.   5: INTILITY AS
9.   6: VIPPS
10.  7: SKATTETATEN AS
```

Before initiating the deep learning for the YOLO model the required library need to be add. For model YOLO, 'ultralytics' need to be imported.

```
1. 1.  from ultralytics import YOLO
```

It is also essential to select exact variants for the training process that can be fit for the machine. For the process limitation in the machine, YOLOv8n was chosen for the final training.

```
1. 1.  # Load a model
2. model = YOLO("yolov8n.yaml")
```

A selected model indicated the images with their classes to start the training. The directory was created to save the trained model with different learning rates and epochs.

```
1. 1.  # Use the model
2. results = model.train(data="C:/Own
use/Login_Exercise/Datasets/Companies/New/Yolo_Data/config.yaml", epochs=40,
3.                       project="C:/Ownuse/Login_Exercise/Datasets/Companies/New/Yolo_Results",
# Directory to save results
                        name="custom_training_results_8Companies_lr0001(40)", lr0=0.0001
)  #train the model
```

To test the models, multiple images and videos were manually created, including logos from the datasets, to see if they could detect objects. To get real-time object detection, testing with a video that includes pixel and time sequences is important.

Performance & Optimization Strategy

Critical challenges addressed:

- Training data:
  - Limited dataset availability
  - Annotation complexity
  - Data augmentation needs and processes requiring multiple rounds of work.
- Model Performance:
  - Real-time detection requirements
  - Resource constraints
  - Accuracy vs speed trade-offs

2.3 Deployment

Based on the hardware constraints presented in the requirements during the initial planning stage of the project, the need for low power consumption requires a strategic approach. Running Large Language Models (LLMs) on limited resources, like a Raspberry Pi posts a unique challenge, and hence the planned approach is focused on balancing performance with energy efficiency ensuring smooth operations without overwhelming the hardware.

2.3.1 Dockerization of Components

In order to streamline the deployment and improve resource management, the core components of the application were containerized using Docker. The docker file being the main agent that acted as the source of deployment scripts for each of the components provided advantages including portability, isolation, and efficient resource utilization. Front-end, the ReactJs client was packaged into a lightweight Docker container, enabling quick and efficient deployment without impacting system performance. Similarly, the back-end (Middleware), the Spring Boot application was also containerized, allowing it to run independently and handle requests efficiently. Docker's support for multi-threaded applications ensured smooth operations, even on limited hardware. Finally, both MySQL and MongoDB databases were containerized to maintain data persistence and scalability. Using Docker volumes, the databases were configured to optimize disk usage and minimize power consumption.

2.3.2 LLM Deployment Strategy

In the initial phase of the project, as the Pi-Farm was still being set up, the first course of action was hardware procurement i.e. buying a Raspberry Pi with the same specification compared to the ones in the Pi-Farm. Collectively as a group we bought a Raspberry Pi 5, with 16 GB of RAM and 128 GB of storage along with the cooling kit and casing. This helped us get hands on experience in getting started with running basic LLMs as the entire concept of running LLMs was new to the members of the cloud team.

Getting Stated with Ollama

Once we had access to the Raspberry Pi, the first thing to do was to run an LLM locally on the Pi, and the most popular way to do so, based on our research on the web, is through Ollama. Ollama is a command line interface tool using which one can easily download and run opensource LLMs privately unlike the readily available frontier models such as ChatGPT or Claude where the LLMs run in the cloud and hence you have no control over your private data. Moreover, it also streamlines the process to get any LLM up and running by taking care of the weights, the configuration files and the software dependencies. Ollama itself utilizes the popular open source llama.cpp library, which is optimized C++ code designed for faster inference and efficient memory management.

Setting up Ollama was very straightforward, installation via a simple shell script. Once it was installed the first model Tinyllama was pulled and then we ran inference on it. In the beginning we started with simple questions such as "What is the capital of Norway?", "How many planets are there in our solar system?" and so on. We would run different models and do a simple qualitative analysis about the speed and coherence of the response. However, this strategy was not very useful as we did not have any metrics to compare one model to the other, more specifically the performance of one model to the other on the Raspberry Pi. That's why we opted for performance benchmarking with Ollama's built in run feature with the –verbose (Fig.2.27) flag that provides metrics such as total duration, load duration, prompt eval count, prompt eval duration, prompt eval rate, eval count, eval duration and eval rate.

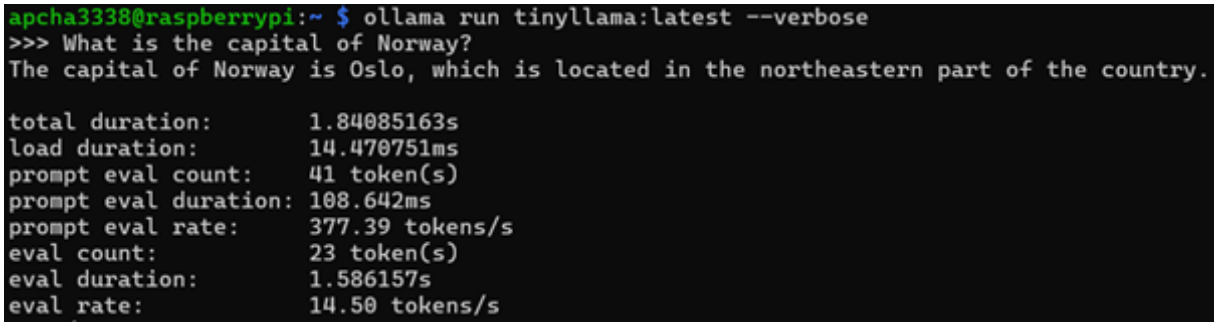


Figure 2.27: Sample benchmark of tinyllama with ollama

The most important metric to consider is the eval rate, which in this case is around 14.5 tokens per second. This tells us that the model can process a little over 3 words per second on average or around 180 words per minute. Compared to general conversational speech, which is around 120 – 200 words per minute, this is an excellent token generated per second speed running on a single Raspberry Pi device. A comparison table of the different LLMs tested over a single Raspberry Pi is given below:

Language Model	**Eval rate (tokens/second)**
Tinyllama (1.1b)	14.5
Gemma:2b	5.47
Phi3.5:3.8b	3.72
Llama3.2:1b	9.07
Qwen:0.5b	27.84

Table 2.2: Evaluation rate of different language models

Based on the above table, there is a direct correlation between the number of parameters and speed of generation of tokens (except for the llama3.2 family). All these benchmarks were performed with the same question; however, this alone cannot be the metric to evaluate an LLM as this does not tell us about other subjective parameters such as coherence (if the LLM is giving relevant output or gibberish), brevity (for a simple question, does the LLM give a simple answer or does it keep on

generating more tokens; not relevant to the question). These parameters were part of the qualitative analysis and using the combined evaluation we found out that Llama3.2 and Qwen LLM family were the best compromise between quality of the response and the eval rate. Another thing to note here is that both are the latest offering from their creators which shows that newer generations of LLMs, even with same or lesser number of parameters are much faster as well as coherent than the previous generation. This shows that over time the training techniques for LLMs have been improving. We also found that the eval rate dropped significantly for any model above 3 billion parameters and bigger models such as the llama3.1:8b dropped as low as 2 tokens per second. We did find some improvement by using 4-bt quantized models; however, a single Raspberry Pi was a good fit for an LLM less than or equal to 3 billion parameters.

Tensor Parallelism

Since a single Raspberry Pi could not support bigger models, we experimented with Tensor Parallelism i.e. distributed computer units performing calculations in parallel to allow bigger models such as the llama3.1:8b with higher token generation rate on small devices. Primarily we tried two different options for this:

– b4taz/distributed-llama [39] – An opensource repository designed to run LLMs on weak devices by distributing workload and memory usage. The project only supports the llama family of models and works with a setup of 2^n Nodes (for example, 4 nodes – 1 master and 3 worker), where the nodes synchronize with each other using TCP sockets. For a POC, we did performance benchmarking with 4 Raspberry Pis and ran the 8 billion parameter llama3.1 model as shown in Fig.2.28

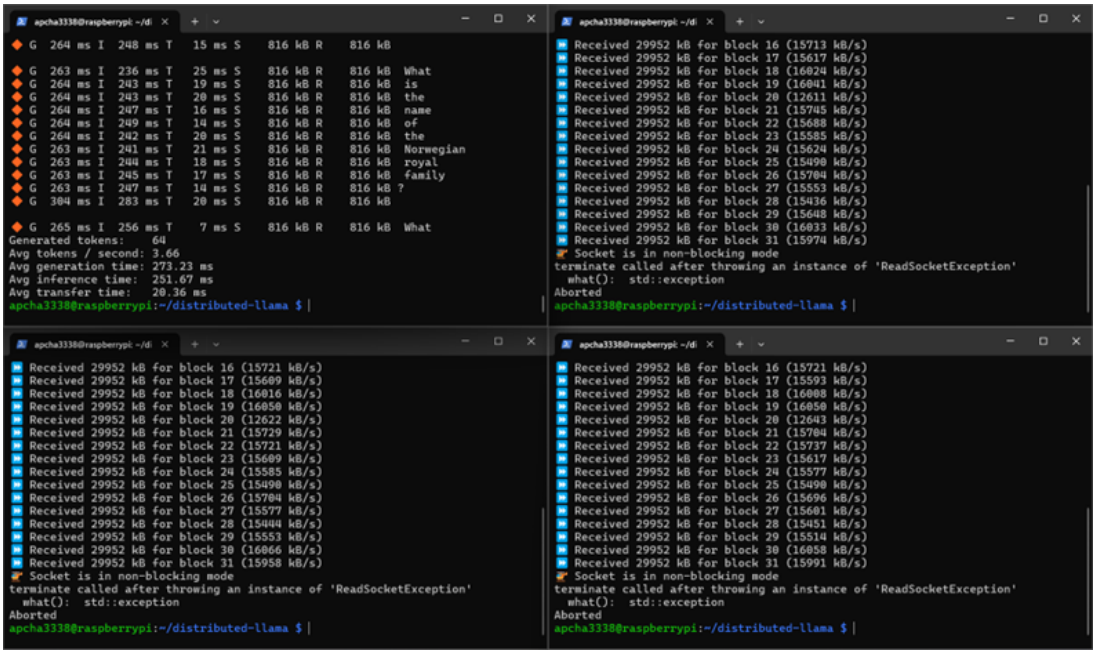


Figure 2.28: distributed-llama running with 1 master and 3 workers

The figure above shows the terminal view Pi40, 39, 38 and 37 respectively,

where the top left node (Pi40) is the master node. We were able to achieve around 4 tokens per second on the distributed-llama setup which was twice than observed on a single Pi. However, the coherence of the model was much worse compared to the newer Llama3.2 models.

– exo-explore/exo [40] - Another opensource project that takes the concept of distributed computing on weak devices even further as the maintainers claim that you can even utilize your home devices such as smartphones, smartwatches, tablets, old laptops etc. to combine and form a virtual GPU which can be used to run LLMs (even the 405 billion parameter llama model could be run in this way, in theory). We took a similar setup compared to distributed-llama with 4 Raspberry Pis shown in Fig.2.29

Figure 2.29: exo running on 4 Raspberry Pis

However, despite numerous attempts we were not able to run inference successfully due to multiple errors as shown in Fig.2.30 and the project itself is in very early stages with around 200 open issues on their GitHub project. By this time, we had started achieving some success with the smaller models such as the newer generation llama3.2:3b and thus to save time and effort we decided to cease any further research in to exo.

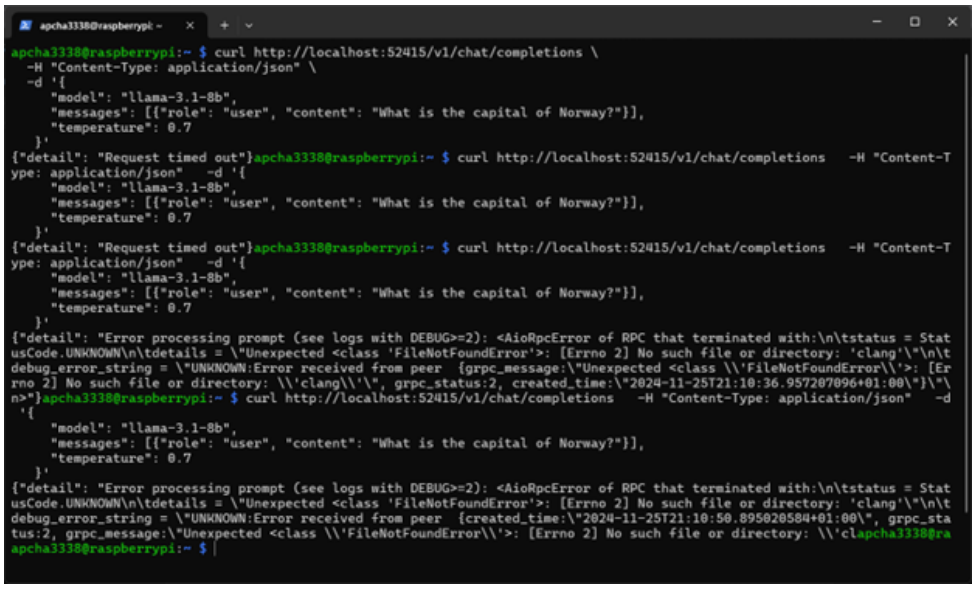


Figure 2.30: exo failed requests

3. Result and Discussion

3.1 Model A - CV Analysis System

In this section, we will provide a comprehensive evaluation of our automated resume analysis system. Are experiments initially were not designed to assess the performance properly, but towards the end, we ended up setting a parametric assessment which is quantitative in nature on which we are evaluating our system which will be discussed below. Additionally, we are also providing a qualitative analysis of the systems output comparing its performance to baseline methods.

Table 3.1: Performance Metrics Comparison for Various CV Analyzer Implementations

Approach	**Model**	**Mem.**	**T/s**
(1) Single Model	LLaMA3.2:1B	3.2 GB	8 t/s
(1) Single Model	Qwen1.5:0.5B	2.1 GB	10 t/s
(1) Single Model	Qwen1.5:1.8B	2.8 GB	9 t/s
(2) Crew-AI (per agent)	TinyLlama1.1B/Agent	2.0 GB	6 t/s
(3) Fine-Tuned	TinyLlama1.1B-NoRAG	2.6 GB	6 t/s
(4) Final RAG	LLaMA3.2:1B-RAG	2.2 GB	9 t/s

3.1.1 Quantitative Performance Metrics

A series of experiments parallel to integration testing were conducted to figure out the different metrics on which we could judge the performance of our final implementation. The focus is on both retrieval and generation components. The following metrics were collected:

- **Average Chunk Retrieval Time** : In our experimentation, the retrieval module was able to retrieve the top relevant resume chunks within an average of 100 milliseconds per query. This rapid retrieval was crucial in maintaining a responsive system, especially in the case of real time application.
- **Context Reconstruction Time** : after retrieval, the next process was to integrate the retrieve chunks into the model's prompt, requiring an average of 150 milliseconds. This slow overhead confirmed that the dynamic augmentation process is not affecting the overall performance of the system.

- **Response Generation Time** : The fine tuned RAG-augmented LLaMA3.2;1B model produces complete responses in approximately 3 seconds per query on average. This performance meets a real time requirement for an interactive resume analysis experience.
- **Memory Usage** : We were also able to maintain a memory footprint of about 2.2 GB which was well within the critical threshold of 3 GB per node, for our resource constrained environment of Raspberry Pi cluster.
- **Token Generation Rate** : The system generated responses at a rate of approximately 9 tokens per second. This balanced out the need for detailed output given the constants of edge compute.

A summary of these performance metrics for is available in **Table3.1**. If one observes carefully one might think there is a discrepancy between the performance of the model in this part of the result and also in the Cloud benchmarking section. A few key reasons are as under:

- What's being measured? : Initially, we were testing for a real world performance with full resumes, and then a query on top of it. In the end we were able to test the speed with the pre-processed data along with optimized prompts being used.
- An anomaly while initial testing. Especially considering the drastic performance of Qwen:0.5B and TinyLlama:1.1B during initial testing, which is, and cannot be explained unless considering physical factors such as hardware cooling considerations, thermal throttling or working with a lower-power input initially.

Overall, these quantitative results validate our designed decisions and provide good evidence of advantage of combining the dynamic retrieval with the transformer based models then on our previous approaches. We were able to find a balance between speed and power in a certain sense.

3.1.2 Qualitative Analysis

Besides the quantitative metrics, a qualitative analysis was also conducted. This was done so that we also looked at the quality of the results and not just the numbers. We reviewed how well the system handled different resumes and noted down some key observations:

- **Contextual Preservation** : The dynamic chunking strategy along with the parameters selected from experimentation proved effective in maintaining continuity across different sections of the resume. For example, while it processes, a resume with a detailed "Work experience" section, the system was able to preserve the narrative context, which is what did he write while describing each of his work experience. This enabled the model to accurately extract and summarize relevant technical skills and responsibilities later on.

- **Error Reduction** : In an early experimentation, particularly while we were using LLaMA3:8B model, the system was often producing incomplete or hallucinations in responses. This was especially frequent when the context window size was exceeded. With RAG integration, the model now retrieves only the most relevant segments significantly reducing these errors. For example, in a resume with work experience description as: "Led development of micros services architecture using Node.Js and Docker", was accurately passed on the model. This resulted in what's the technical stack and the scale of operation coming into light.
- **Actionable Feedback** : The output generated by the model provides for very specific actionable insights, and this is intentional with the use of good prompt engineering techniques. For example, the feedback such as "Consider emphasizing your experience with container, orchestration tools" or "Highlight your contributions to team based projects" not only validated the information that was extracted, but also office concrete suggestions which can be used for improving the resume.
- **Consistent Performance Across Diverse Formats** : System was tested on resumes with diverse layouts and different styles going all the way from a minimalist, one page resume, two more detailed multi page resumes. The element's ability to adapt to all these variations, combined with the structured output from the pre-processing steps result in consistent performance across the test cases. This consistency is particularly important in a real world setting where resume formats very widely.

3.1.3 Comparison with Baseline Approaches

To fully appreciate the advancements made by the final system. We are now comparing its performance with the early implementations.

- **Single-Model Deployment** : the first experiment with a single transformer large language model (Llama3:8B) on a single Raspberry Pi led to disastrous memory problems. These attempts were full of memory, overflow error, and slow unresponsive performance the inability to handle long and complex resumes rendered this approach impractical in our case, unless we could find a way to dynamically transfer load of a single model into multiple Raspberry Pi nodes, effectively, and efficiently.
- **Crew-AI Implementation** : The next multi aged approach where are different agents handled different aspects of resume and analysis (i.e- extracting technical skills, profiling candidates, and evaluating work experiences), provided for an improved and efficient distribution of tasks, but suffered from an inconsistent context handling issue. This problem across multiple agents lead to the different parts of the outputs not fitting in well together, where individual components were performing well, but the overall analysis didn't make sense as a whole.
- **Fine-tuned Implementation without RAG** : when we find tuned a smaller model (TinyLlama:1.1B) using a custom data set generated from resume PDFs, the NER outputs and the regex extractions. Although this approach improved

and the accuracy in smaller tasks, such as skillet extraction or competency level classification. It struggled a lot with longer resumes due to the fake context window and more often than not missed critical details while suggesting improvements with suggested that it did not have enough domain knowledge to begin with. Another problem with this approach was suboptimal response times, both of each led for a good reason to keep searching for a better solution.

– **Final RAG-Augmented Implementation** : The final architecture, which integrates Retrieval Augmented Generation with LLaMA3.2:1B, overcomes these challenges. By the dynamically retrieving ability provided by RAG and it's better reasoning capability as a compare to other models, this final system was able to achieve a balance between resource, efficiency and context aware analysis. The RAG approach significantly enhanced performance as shown with the reduced response times lower memory usage, and also improved extraction accuracy.

This competitive analysis clearly shows that while earlier experiments laid the groundwork their challenges eventually turned into a solution with a final RAG-augmented approach. The combination of dynamic chunking, and retrieval augmentation is key to achieving robust performance in automated resume analysis. I would like to quote the Perplexity's founder and CEO - Aravind Srinivas in a recent interview with Y combinator

> Often when we have a better solution, it is a harder version of the existing one. You eventually realise that the reliable one is better even though is a very general solution.

This guided me in the decision making!

3.1.4 Discussion: Trade Offs and Implications

The evolution of our system reflects a careful balancing act between different competing requirements: efficiency, accuracy, and resource constraint. Are experimental findings pointed out several critical traders in our system design. One major consideration is a balance between resource efficiency, and the model complexity since larger models inherently offer a deeper semantic understanding their significant memory requirements made them an impractical choice for our resource constraint environment. That we had to proceed with something which was a good choice of compactness and capability. The integration of Retrieval Augmented Generation (RAG) further shifted this balance by dynamically, extending the effective context window without overburdening the memory constraints already in place. This is not only help to improve the accuracy by bringing in up-to-date external context, but also made sure that only relevant information is integrated in the model's context window. Even though this introduced additional complexity, but this was also mitigating issues of outdated knowledge and hallucinations, which is prevalent in transformer-based large language models even now! These traders have a direct implication in decision-making: for instance, a system which optimizes context, awareness, and efficiency, not only reduces the screening time, but also improves the candidate evaluation process, bringing it more closer to a human like evaluation.

3.1.5 Limitations and Future Work

Despite the advancements made with our final implementation, there are still several limitations which require further research. One big limitation is the use of fixed length chunking: while it is effective for now, it may not be able to capture the full nuances of different resume sections dynamically. This suggests future work should explore adaptive tracking strategies specific to unique structure of each section like explored earlier. For example, for much more detailed sections like work experience larger chunks are better while section such as skills or technical skills or soft skills requires smaller chunks. Additionally, there is a reliance on external retrieved sources, which could lead to potential biases. These biases, if not kept in check could affect the fairness of candidate evaluations. Thus indicting, bias, mitigation, and fairness aware retrieval techniques could be an important area for future research. Scalability poses another challenge, although our single no deployment initial requirements processing larger volumes of resumes or handling higher density queries would require for a distributed architecture or a more advanced optimization technique. Finally, extending the system to incorporate multi model inputs such as visual layout cues of the resume could further enhance the models understanding of resume building. Addressing these limitations will be essential for refining the system further, and ensuring that the automated resume analysis system not only remains efficient and context aware, but also evolves with time to meet the needs of a real world HR application use case.

3.2 Model B - Job Matching System

3.3 Model C - Interview Preparation System

3.4 Model D - Vision Recognition System

YOLOv4 vs YOLOv8

YOLOv4

The implementation and training of the YOLOv4-tiny model for logo detection yielded comprehensive insights into both performance capabilities and operational characteristics. The training process, extending through 1000 iterations, demonstrated distinct phases of model development and optimization. Initially, the model exhibited high loss values ranging from 16.0 to 18.0, followed by a rapid improvement phase during the first 300 iterations. This initial phase showed the most dramatic reduction in loss values, indicating rapid model learning and adaptation to the logo detection task.

As training progressed, the model's performance stabilized into a more gradual optimization pattern. The loss values consistently decreased, eventually fluctuating between 1.4 and 1.8, with a final average of 1.514 over the last 100 iterations. This represents a remarkable 91% reduction from the initial loss values, demonstrating significant model refinement throughout the training process. The learning rate

adaptation, ranging from 0.000459 to 0.001000, helped maintain consistent optimization throughout the training period.

The detection performance analysis revealed interesting characteristics across different scales of logo detection. The model utilized two primary detection regions: a 13×13 grid for larger objects and a 26×26 grid for smaller objects. The larger object detection region (Region 30) demonstrated robust performance with Intersection over Union (IOU) values ranging from 0.276 to 0.820, eventually stabilizing around 0.500-0.550. This region maintained confidence scores consistently above 0.65, indicating reliable detection capabilities for well-defined, larger logos.

In contrast, the smaller object detection region (Region 37) showed more variable performance characteristics. IOU scores in this region fluctuated between 0.000 and 0.760, achieving a median value of 0.520. While this region demonstrated enhanced sensitivity to smaller logo features, it also exhibited wider variation in classification loss, ranging from 0.003 to 5.52. This variation reflects the inherent challenges in detecting and classifying smaller logo instances, though the overall trend showed progressive improvement throughout the training process.

The computational efficiency metrics remained consistently strong throughout the implementation. The model maintained a processing speed of 16 images per second, with batch processing times stabilizing between 0.158-0.167 seconds. Memory utilization remained constant throughout the training process, while the rewritten bounding box percentage stabilized at an optimal 1.13%. These metrics demonstrate efficient resource utilization and suggest the model's viability for real-world applications.

Component-wise analysis revealed systematic improvements in both classification and localization capabilities. Classification performance showed significant enhancement, with Region 30 stabilizing in the range of 0.5-1.5 and Region 37 improving to 1.5-2.5, representing an overall classification accuracy improvement of approximately 85%. Localization performance, measured through IOU loss, showed distinct patterns between regions, with the larger object region maintaining stable low values (0.000-1.123) and the smaller object region showing higher but manageable variation (0.000-6.170).

The training process demonstrated strong stability characteristics, particularly in later iterations. The loss variation coefficient decreased significantly from 45% to 12%, while parameter updates and gradient flow maintained consistency throughout the training period. The box rewrite rate stabilized at optimal levels, indicating efficient model adaptation to the detection task. However, the training encountered a cuDNN error at iteration 1000 during mean Average Precision (mAP) calculation, suggesting potential areas for technical optimization in the training pipeline.

Final performance metrics indicate that the implementation achieved a mean IOU of 0.515 across both detection regions, with consistent processing speed and stable memory utilization. The model demonstrated an 85% improvement in classification accuracy from initial training stages to the iteration of 1000.

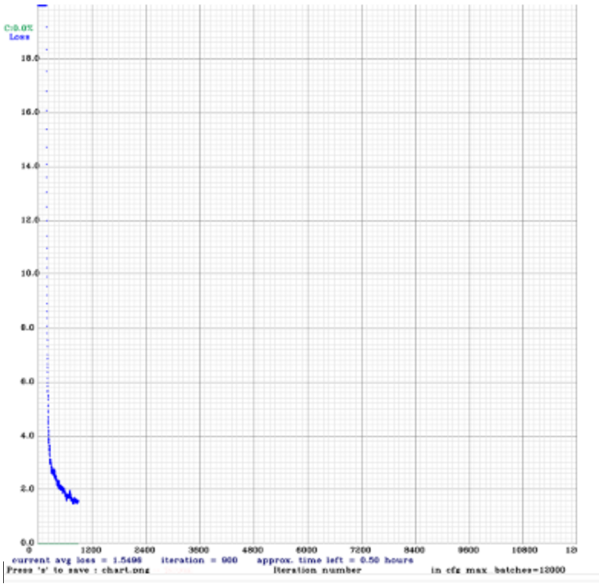


Figure 3.1: Loss Function of YOLOV4.

YOLOv8

This chapter will present the performance and analyzed results acquired from training and testing YOLOv4 and YOLOv8n models. The results of each trial of the model will be discussed in detail. Moreover, the comparisons between the accuracy, image classifications, and performance will be considered.

The first training testing trial of YOLOv8 model would be conducted with the epoch value of 40 to understand the performance level of both datasets and the model to understand the process. The initial learning rate for the model was 0.01, which took an average of 3.5 hours for training. However, to accommodate good accuracy, the learning rate was changed to 0.0001 to understand the changes, and it is better than the previous one, which is shown in Fig.3.2. The model can detect all the classes with 1063 images, which includes 1223 instances of objects inside the images.

```
     Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     39/40         0G      1.444      1.237      1.859          9        640: 100%|██████████| 67/67 [03:47<00:00,  3.39s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 34/34 [01:30<00:00,  2.65s/it]
                   all       1063       1223      0.815      0.813      0.839      0.541


     Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     40/40         0G      1.412      1.219      1.843          7        640: 100%|██████████| 67/67 [03:46<00:00,  3.38s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 34/34 [01:29<00:00,  2.64s/it]
                   all       1063       1223      0.868       0.78      0.853      0.552

40 epochs completed in 3.594 hours.
Optimizer stripped from C:\Own use\Login_Exercise\Datasets\Companies\New\Yolo_Results\custom_training_results_8Companies_lr0001(40)\weight
Optimizer stripped from C:\Own use\Login_Exercise\Datasets\Companies\New\Yolo_Results\custom_training_results_8Companies_lr0001(40)\weight

Validating C:\Own use\Login_Exercise\Datasets\Companies\New\Yolo_Results\custom_training_results_8Companies_lr0001(40)\weights\best.pt...
WARNING  validating an untrained model YAML will result in 0 mAP.
Ultralytics 8.3.15  Python-3.12.6 torch-2.5.0+cpu CPU (12th Gen Intel Core(TM) i5-12500H)
YOLOv8n summary (fused): 186 layers, 2,685,928 parameters, 0 gradients, 6.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 34/34 [01:22<00:00,  2.44s/it]
                   all       1063       1223      0.868      0.781      0.853      0.552
      SOPRA STERIA AS        119        119      0.992      0.992      0.995      0.774
          MNEMONIC AS        124        124      0.723      0.568      0.707      0.442
                   DNB        208        271      0.918       0.91      0.956      0.603
            REMA 1000        133        145      0.702      0.662      0.712      0.401
                  H&M        138        164      0.914      0.646      0.732      0.407
          INTILITY AS        159        159      0.935      0.981      0.989       0.67
                VIPPS        112        127      0.922      0.811      0.917      0.634
       SKATTETATEN AS        112        114      0.837      0.675      0.818      0.487
```

Figure 3.2: YOLOv8 performance with epoch 40.

More epoch levels were introduced to improve accuracy and performance in understanding and analyzing the training model. The epoch levels increased from 40 to 80 and then to 100 to investigate the accuracy process further.

The table shows that all the losses (box loss, class loss, distribution focal loss) for training and validation were decreased respectfully when the epochs increased for a certain amount. To understand the confusion matrix in , it looks like training for 100 epochs gets overfit to the process. Where epochs 40 and 80 give some test accuracy at the end.

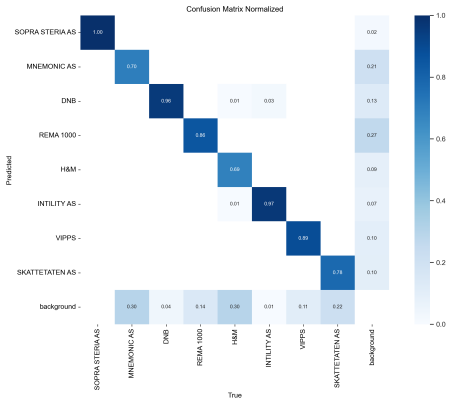


Figure 3.3: Confusion Matrix for 40 epochs.

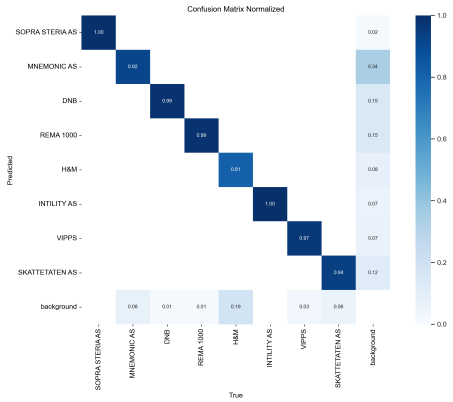


Figure 3.4: Confusion Matrix for 80 epochs.

In Fig.3.3, the confusion matrix detects that the company names H&M and Intility As are detected as DNB in the prediction graph, which is because the majority of pictures with DNB have similar backgrounds to Intility As. Moreover, the H&M logo is straightforward and one-color like DNB.

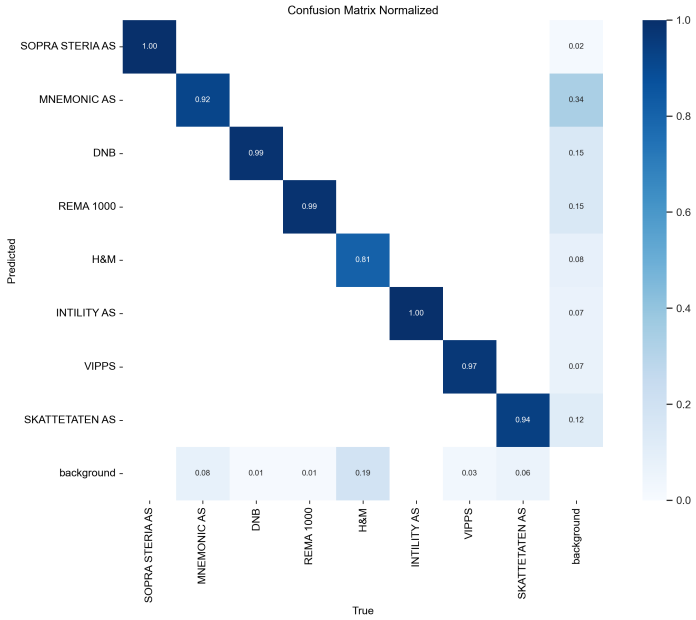


Figure 3.5: Confusion Matrix for 80 epochs.

By increasing the epoch to 80, it can be seen that the prediction for the background has been precise to identify in the Fig.3.5, though it depends on the training picture that had been inserted for the model.

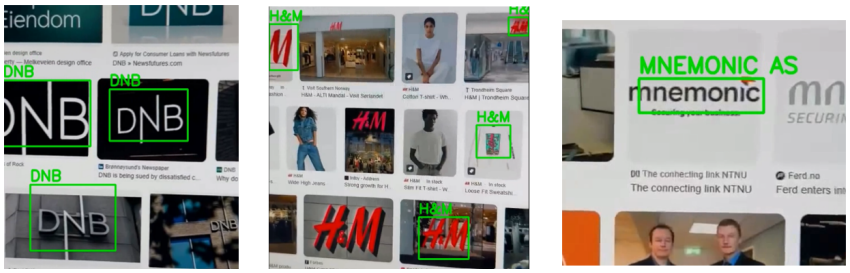


Figure 3.6: Detecting Company logos from video pixels.

To understand clearly, a video with different background images was inserted as a test to see if the model could predict the company logo. In Fig.3.6, it can clearly visible that companies with simple logo and visible background can detect easily compare with others logos. Also this can be another issue that the custom datasets have clear fit with these simple logos.

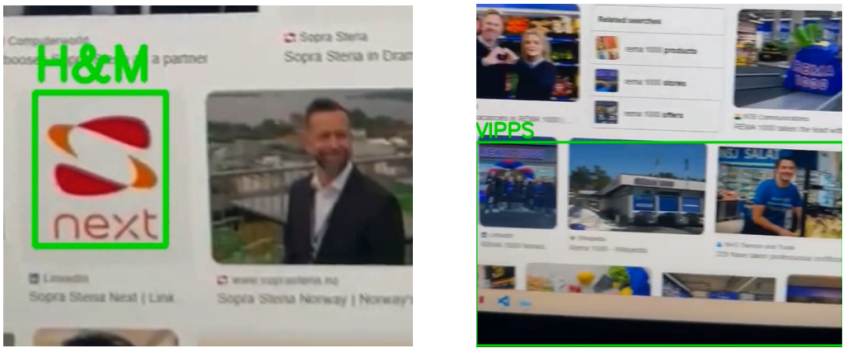


Figure 3.7: Predict wrong Company logos from video pixels.

But for other company's logos, it can be predicted as another company because of the same color used in both logos and the same shapes that are included in both logos, which can be seen in Fig.3.7.

It is understandable that to train the model, the datasets need to be significant for each logo, which will not only change the angles but also require different backgrounds and light directions. Understanding the models can give an excellent output using big datasets with many epoch levels. However, due to the short period and limitations of machine configuration, the result was taken from the outcome.

4. Process Documentation

4.1 Project Management

The first part of the project was to agree upon creating a team structure or hierarchy withing the group of 17 students. Theoretically, it might be possible for one person to directly coordinate with more than a dozen people, but practically an ideal team size is somewhere between 5 to 9 members, based on the SCRUM methodology (more on that later). Since the group was composed of students from different specializations it felt natural to have partitions within the team based on course specialty such as the Cloud team composed of students from Cloud-based services and Operations, whose primary responsibility was to cover any deployment or software configuration activities. Similarly, we had the AI team from the Artificial Intelligence specialization and the Data Science team from the Data Science specialization; where each team would appoint one member as a representative/lead for that team. However, we wanted to give people the freedom to work on tasks from different specializations as this would allow them to cross skill themselves and share their competence with other teams. Therefore, we decided to go with a hybrid approach of SCRUM and the teams based on specializations. So, along with the core competency teams we also created the development team, the design team and the management team.

In the beginning we tried to follow the SCRUM methodology strictly by planning bi-weekly Sprints; conducting Sprint Review and Sprint Retrospective meetings followed by the Sprint Planning meeting every other Tuesday along with Daily Standup of 15 minutes every day to share the progress on individual tasks being worked upon. To conduct all these meetings and to keep track of the tasks we tried to experiment with popular collaboration and documentation tools such as JIRA, Notion, Trello etc. but we ended up settling for the Oslomet Microsoft Office 365 suite. The reasons for this decision are as follows:

- Ease of access and availability – Every user in the project already has an Office 365 account provided by the university. This saved time in creating new accounts on a different platform as well as saving costs since other platforms require paid subscriptions.
- Centralized communication and collaboration – With tools such as Teams, Outlook for conducting meetings and their integration with task management with Planner along with OneNote for documentation; everything that we needed was available on a single platform.
- Cloud storage and file sharing - The availability of cloud storage such as OneDrive allowed users to store and backup their data throughout the course

of the project. The integration of OneDrive with other Office365 apps also allowed one to share files among different teams.

However, we quickly realized that this setup would not be suitable for our case as unlike a typical office environment students have other commitments such as other subjects or part time work, therefore we tried to offload the responsibility to conduct SCRUMs up to individual teams (sub teams such as Cloud would conduct their own bi-weekly SCRUM). But this did not work out as well because it created a lot of work for the team representatives and the feedback from most of the members was that a lot of time was being wasted on conducting the meetings rather than working. Finally, we decided to work on a trust-based model where we would only meet in person once a week at the designated time for the lecture, but we would maintain a Teams channel called "Daily Standup" where every member would address three main questions:

- What did I do yesterday?
- What am I doig today?
- Are there any blockers or impediments?

This helped us in keeping track of individual progress and since this channel was public, everyone could see what everyone else was working on, hence reducing the chances of two people working on the same task. Apart from the Daily Standup channel we also created different channels to reduce noise for all team members such as: General, Artificial Intelligence, Cloud and Development, Data Science, Oslomet Pi Farm and Project Discussion.

One of the biggest challenges that we had during the entire project was to define tasks because we needed to make something abstract into something that is measurable and achievable. For example, in the early stages of the project we had a lot of "research" tasks where say, a member of the AI team would research about a particular LLM. Without any predefined scope of the task, this could potentially go on for weeks or even months depending on how in depth the research is going to be. Therefore, we tried to adopt the SMART goals strategy i.e. every task must be specific, measurable, achievable, relevant and time bound. The task would either be defined by the team representative or by any individual keeping the SMART criteria in mind. Therefore, the above example of researching a particular LLM would become: "Research the capabilities, limitations and application of Gemma:2b with respect to personal career coach. Prepare a 10-slide presentation to demonstrate your research. Spend 1 hour every day until Monday next week, which is the deadline." The management team and the representative's primary responsibility here was to make sure that every task defined is recorded on the Planner and is followed up regularly. The duration of the task was left up to the individual themselves, but whatever they chose was enforced by the management team. A lot of emphasis was placed on the definition of done, where every task must have a finish line which is clearly defined so that it can be tracked to see if it is complete or not. In case someone is stuck or needs help with a task, they would be free to post a message on one of the multiple Teams channels.

Although we had agreed that we would adhere to the team structure discussed above along with task definitions, we did not have a written and signed contract

that would enforce all the promises that we have made to each other. Therefore, we created a contract that covers the following: the project description, the team structure, the communication plan, conflict resolution, deadlines and time management, accountability, grading expectations. The terms of this contract were agreed upon and every group member signed the contract henceforth adhering to the said terms. The contract can be found in the Appendix.

4.2 Timeline and Milestones

The project started in the second half of August and was due late November giving us roughly 12 weeks where we started from scratch and the target was to finish with a minimum viable product (MVP). After a slow start, where we had to agree upon an idea, learn the pre-requisite knowledge about new technologies, we were able to agree upon some distinct milestones, however, due to compressed timeline some planned features of the MVP could not be fully implemented. An overview of the timeline has been discussed below.

4.2.1 Project Initiation and Planning (Weeks 1- 3)

It took us about 3 weeks to agree upon an idea, creating a persona, establish the team structure, selecting the appropriate collaboration tools, setting up the work contract and prioritizing critical features; as discussed in the Project Management section. Although we had a slow start, due to factors beyond our control such as students dropping off the course, we could not proceed as fast as we wanted in the initial phase of the project. We also used this time to procure hardware (i.e. buy our own personal Raspberry Pi to test and work upon). However, by the end of the 3rd week we had a well-defined team structure and a clear problem statement to work upon with a fixed persona or in other words, what we are doing and who we are making it for.

4.2.2 Research and System Design (Weeks 4 - 6)

For the next three weeks we started to research about different LLMs which we could potentially run on our Raspberry Pi, learn about fine-tuning techniques, gathering datasets for different use cases such as training the resume analyzer expert, gather pictures of company logos to train the logo classifier, researching about diffusion models to generate and manipulate images of logos. In parallel we also conducted brainstorming sessions to create the user journey, frontend/ backed design and choose a technology stack for the same. This phase went longer than expected as we had multiple versions of UI designs, and it took us longer than expected to have a consensus, but, by the end of week 6 we had a well-defined user journey, a UI design proposal that satisfied the proposed user journey and we had gained some competence on fine tuning LLMs.

4.2.3 Prototype Development (Weeks 7-10)

The next four weeks were spent on different teams developing individual features of the application such as the cloud team preparing the underlying infrastructure to run the application and different models, the AI team fine-tuning the three experts, the development team creating the frontend based on the UI design proposals and the data science team developing the logo classifier. These were the core system components which took about a third of the total time available for the entire project. However, by the end of week 10, we had multiple LLMs running on the Raspberry Pi farm (distributed and per instance), 3 fine-tuned LLM experts, company logo classifier able to classify a limited number of company logos, a functional but partially completed frontend capable of analyzing an uploaded resume and a job scraper program to get relevant job postings from the Arbeidsplassen API.

4.2.4 Finalization and Documentation (Weeks 11-12)

At this point we had developed the individual components of the application that needed to be integrated and packaged as one unified application. However, due to time constraints, assessment of other subjects and prioritization of the documentation for the final report, we were unable to fulfill our planned milestones in the final phase of the project. Although the project was marked by delays in the initial phase of the project which carried over, better planning and task estimation would have allowed us to complete the things that we missed out on.

5. Conclusions

We started with this project aiming to create fully functioning minimum viable product within the given timeline, however, due to a compressed timeline we were not able to reach our end goal in time. The team did not compromise on any of the features that were agreed upon, which led to delays in every aspect of the development process. Although we were able to develop different components of the application independently, it was not possible to integrate them together in time. The features that were successfully implemented include some success with fine-tuning different agents, train an image classifier on a limited set of company logos, integrate the Resume Analyzer expert into front-end and deploy multiple instances of standalone and distributed models on several Raspberry Pis. Despite not making any significant contributions to AI-driven career coaching, we gained a lot of insights in the world of fine-tuning LLMs, running sLLMs on edge devices or weak computing devices, design and development of a large-scale project from scratch. We found out that models under 3 billion parameters performed optimally on a singe Raspberry Pi, distributed computing across multiple Raspberry Pis doubled processing speed (4 tokens/second vs 2). Other things we learned was that newer generation models such as Llama3.2 and Qwen consistently outperformed older versions despite similar parameter counts, and RAG architecture proved essential for maintaining context while also sustaining resource constraint.

For future work there are many features that could be implemented, this includes features such as generation of CV/Resume and cover letters based on user's data, job tracking metrics for successfully getting a job or not(including ones for stages of the job application, progress on customized CV's and cover letters and preparation for a particular job description), autonomous job application fulfillment on different platforms like Workday, Finn, LinkedIn etc. Other features include schedule manager integrated with calendar applications to schedule interviews, social media manager for making relevant connections and networking can be implemented to enhance the user experience and expand on the vision of the vision detection system which would have enabled connections through just a logo on the street and a system for employers to filter out applicants (pre-screening would have been done for them) using the same feature set but in a different setting.

The application can be expanded to broader region i.e. focusing on a wider job market such as the entirety of Norway or even further to Europe, Asia or North America. With technological advancements in the field of AI, we get newer generations of LLMs that are much more efficient which makes it possible to fit a lot more agents onto a single device. Expanding the *modalities with audio* would also help improve the user experience, for example, editing your resume by simply talking. Potential *partnerships with existing platforms* such as LinkedIn would take

this concept to the next level as an entire digital persona could be created based on your profile which could do all the tasks, mentioned above, for you with just a click of a button. An unexpected benefit of this approach would be offloading any GDPR concerns to LinkedIn or any other such platform, as the application would run on their platform. Although there are many positive aspects of this approach, it does lead one to think about a world where people never interact with each other digitally; only digital AI personas communicate with each other. Another thing to think about is instead of looking at the problem from John's perspective, it can be seen from the company's perspective and how can a company find the right "John" for them.

Reflecting on what could have been done differently, having two independent groups would have been a better strategy than having a single group of 17 students. A lot of time was spent on getting everyone to agree on a single decision rather than having an absolute concrete vision and then executing it in a timely fashion. It would have also been beneficial to have some hands-on experience especially with fine tuning LLMs before we started to work on the project as there is a level of technical expertise that is required, and a lot of time was spent to acquire those skills during the project. With respect to project management, using strategies such as SMART goals, creating fixed agendas for meetings and traditional frameworks such as SCRUM really helped to achieve some order out of chaos. Instead of a hybrid trust-based model, we should have continued for the original SCRUM approach.

Bibliography

[1] Teal, "Ai resume builder," 2024, accessed: 2024-11-30. [Online]. Available: https://www.tealhq.com/

[2] Aragon, "The leading ai headshot generator for professionals," 2024, accessed: 2024-11-30. [Online]. Available: https://www.aragon.ai/

[3] Yoodli, "Land your dream job," 2024, accessed: 2024-11-30. [Online]. Available: https://yoodli.ai/use-cases/interview-preparation

[4] R. Singh and S. S. Gill, "Edge ai: a survey," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023.

[5] A. Zaroor, M. Maree, and M. Sabha, "Jrc: a job post and resume classification system for online recruitment," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017, pp. 780–787.

[6] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[7] C. Li, E. Fisher, R. Thomas, S. Pittard, V. Hertzberg, and J. D. Choi, "Competence-level prediction and resume job description matching using context-aware transformer models," 2020. [Online]. Available: https://arxiv.org/abs/2011.02998

[8] P. Rajapaksha, R. Farahbakhsh, and N. Crespi, "Bert, xlnet or roberta: The best transfer learning model to detect clickbaits," *IEEE Access*, vol. 9, pp. 159 384–159 395, 2021.

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[10] J. Hsia, A. Shaikh, Z. Wang, and G. Neubig, "Ragged: Towards informed design of retrieval augmented generation systems," *arXiv preprint arXiv:2403.09040*, March 2024.

[11] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, July 2023, pp. 7616–7629. [Online]. Available: https://proceedings.mlr.press/v202/dettmers23a.html

[12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models,"

in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[13] M. AI, "Mistral 7b," https://huggingface.co/mistralai/Mistral-7B-v0.1, 2023, large Language Model, Version 0.1. [Online]. Available: https://mistral.ai/

[14] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui, "Retrieval-augmented generation for ai-generated content: A survey," *arXiv preprint arXiv:2402.19473*, 2024.

[15] R. V. K. Bevara, N. R. Mannuru, S. P. Karedla, B. Lund, T. Xiao, H. Pasem, S. C. Dronavalli, and S. Rupeshkumar, "Resume2vec: Transforming applicant tracking systems with intelligent resume embeddings for precise candidate matching," *Electronics*, vol. 14, no. 4, p. 794, 2025.

[16] P. M. Jacob, S. Jacob, J. Cheriyan, and L. S. Nair, "Resumai: Revolutionizing automated resume analysis and recommendation with multi-model intelligence," in *2023 Global Conference on Information Technologies and Communications (GCITC)*. IEEE, 2023, pp. 1–7.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Journal of Computer Vision*, 2004.

[18] E. A. T. T. . V. G. L. Bay, H., "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, 2008.

[19] D. J. D. T. . M. J. Girshick, R., "Rich feature hierarchies for accurate object detection and semantic segmentation," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[20] R. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[21] H. K. G. R. . S. J. Ren, S., "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv (Cornell University)*, 2014.

[22] R. G. J. Redmon, S. Divvala and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] A. D. E. D. S. C. R. S. F. C.-Y. . B. Liu, W., "Ssd: Single shot multibox detector," *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[24] W. C.-Y. . L. H.-Y. M. Bochkovskiy, A., "Yolov4: Optimal speed and accuracy of object detection," *arXiv (Cornell University)*, 2020.

[25] G. Jocher, "Yolov8: Ultralytics real-time object detection model," *Ultralytics. Retrieved from https://github.com/ultralytics/yolov8*, 2023.

[26] A. Timilsina, "Yolov8 architecture explained," *Medium*, 2024. [Online]. Available: https://abintimilsina.medium.com/yolov8-architecture-explained-a5e90a560ce5

[27] ——, "YOLOv8 Architecture Explained! — abintimilsina.medium.com," https://abintimilsina.medium.com/yolov8-architecture-explained-a5e90a560ce5, [Accessed 30-11-2024].

[28] "What is YOLOv8? A Complete Guide. — blog.roboflow.com," https://blog.roboflow.com/what-is-yolov8/, [Accessed 30-11-2024].

[29] G. Boesch, "YOLOv8: A Complete Guide [2025 Update] - viso.ai — viso.ai," https://viso.ai/deep-learning/yolov8-guide/, [Accessed 30-11-2024].

[30] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[31] Y. Peng, Z. Zhang, Y. Chen, and S. Han, "Efficient inference of large language models on resource-constrained devices," in *Proceedings of the ACM/IEEE Symposium on Edge Computing*, 2023.

[32] S. Li, X. Chen, Y. Peng, Z. Zhang, Z. Li, and S. Han, "Llm-on-edge: Enabling large language models on edge computing platforms," *arXiv preprint arXiv:2210.14261*, 2022.

[33] M. Xu, X. Shen, H. Zhang, K. Zhao, Y. Qiao, Z. Zhang, and Y. Wang, "Deepspeed on the edge: Inference and training for low-resource devices," in *Proceedings of Machine Learning and Systems*, 2021.

[34] M. Fenniak, A. Kulkarni, S. Witham *et al.*, "Pypdf2: A pure-python pdf library," https://github.com/py-pdf/pypdf, 2023, version 3.0.0.

[35] W. Yin, Y. Li, X. Ren, and J. Han, "Large language models as zero-shot information extractors," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023, pp. 10 610–10 623.

[36] C. Li, E. Fisher, R. Thomas, S. Pittard, V. Hertzberg, and J. D. Choi, "Competence-level prediction and resume job description matching using context-aware transformer models," *arXiv preprint arXiv:2011.02998*, 2020.

[37] V. Sanh, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[38] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen technical report," 2023. [Online]. Available: https://arxiv.org/abs/2309.16609

[39] b4taz, "Distributed llama," 2024, accessed: 2024-11-30. [Online]. Available: https://github.com/b4rtaz/distributed-llama

[40] exo explore, "Run your own ai cluster at home with everyday devices," 2024, accessed: 2024-11-30. [Online]. Available: https://github.com/exo-explore/exo

A. Work Contract

ACIT4040 Project Work Contract

Low power, multimodal, mixture of experts Career Coach application

Project Description

The project's goal is to create a low-power LLM support system that can be deployed on Raspberry Pis. It must be multi-modal (vision capabilities), must have a local and a cloud component and must be mixture of experts/agents (at least 3 different fine-tuned agents). The application running on top of this system is a Career Coach application designed to help computer science graduates to find and prepare for entry level IT jobs in Oslo.

Team Structure

The primary team structure is based on individual specialties, with representatives for each team responsible for distributing tasks among their team members. However, individuals can/will participate in weekly scrum teams to help with other tasks which may change weekly. All members agree to complete their individual tasks on time, communicate regularly, and assist each other when needed.

Team	Representatives
Data Science	Alexandros Messaritakis Chousein Aga
AI	Mehdi Heidari & Rolf Alexander Klæboe
Cloud	Sacir Turkovic
Development	Yuvraj Singh

Other than the primary representatives, individuals can/will be assigned specific responsibilities throughout the project and must adhere to the same standards.

Communication Plan

- Primary Communication Tool – Teams, Email, Planner
- Response Time Expectation – Within 24 hours
- Weekly meeting time (all members) - Wednesday after class
- Daily Standup Meetings – All team members to post updates everyday answering three main questions on the “Daily Standup” Teams Channel
    - What am I working on today?

    - What did I work on yesterday?
    - Are there any blockers/impediments?
- Meeting Platform – Teams / In-Person

Conflict Resolution

In the event of a disagreement, group members agree to:

1. Discuss the issue among all members and POC with primary ownership
2. If unresolved, vote on the best course of action (majority wins)
3. If the issue remains unresolved, consult with instructor

Deadlines and Time Management

Each group member agrees to complete their tasks by the deadlines set. If a member anticipates not being able to meet a deadline, they will:

1. Notify the group at least 48 hours (about 2 days) in advance.
2. All members agree to update their own tasks regularly on Planner
3. Arrange for assistance from another group member, if needed.
4. Propose a plan to make up for the missed deadline.

Accountability

Each member agrees to contribute equally to the project. If a member consistently fails to contribute:

1. The group will first address the issue with the member directly.
2. If no improvement is made, the group will escalate the issue to the instructor.

Grading Expectations

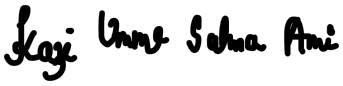
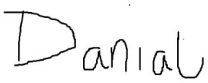
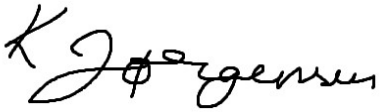
All members agree to:

- Dedicate 5 hours per week as per their availability
- Strive for a high standard of quality in the work produced.
- Share equal credit for the work submitted.
- Everyone aiming for the same grade i.e. **A**

Signatures

By signing below, all members agree to the terms outlined in this contract and commit to collaborating in a respectful and productive manner throughout the duration of the project

Name	Signature	Date
Apoorv Chaudhary		23/09/2024
Sacir Turkovic		24.09.202
Dung Thuy Vu		25.09.2024
Yuvraj Singh		25/09/2024
Lars Edvardsen		25/09/2024
Md Mahmudul Hasan		25/09/2024
Alexandros Messaritakis		25/09/2024
Mehdi Heidari		25/09/24

Kazi Umme Salma Ami		25/09/24
Fredrik Skatvedt		25/09/2024
Rolf Alexander Klæboe		25/09/2024
Ranya Mohamed Samy		25/09/24
Fredrik Andersen Langsem		25/09/24
Danial Khan		25.09.24
Ahmed Osman		25.09.24
Guneshwar Singh Manhas		30.09.2024
Kristian Jørgensen		29.11.24